UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Didka Dimitrova Birova

Primerjava orodij za vizualizacijo omrežij

DIPLOMSKO DELO

UNIVERZITETNI ŠTUDIJSKI PROGRAM
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: izr. prof. dr. Marko Bajec

Ljubljana, 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Pri analizi omrežij nam vizualizacija omrežij olajša njihovo razumevanje ter omogoča dodatno interpretacijo rezultatov analize. S prikazom omrežja imamo vpogled tako v njegovo globalno strukturo, kot tudi v pomen posameznih vozlišč v omrežju. Diplomska naloga naj obsega pregled obstoječih paketov za vizualizacijo omrežij. Njihova učinkovitost in uporabnost naj se primerja na realnih omrežjih različnih velikosti. Opisani in prikazani naj bodo primeri uporabe posameznih knjižnic ter njihove prednosti in slabosti.

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisana Didka Dimitrova Birova,

z vpisno številko 63060403,

sem avtorica diplomskega dela z naslovom:

Primerjava orodij za vizualizacijo omrežij

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelala samostojno pod mentorstvom izr. prof. dr. Marka Bajca
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 23.4.2014 Podpis avtorice:

Zahvala

Zahvaljujem se mentorju izr. prof. dr. Marku Bajcu kot tudi as. uni. dipl. ing. Neli Blagus za pomoč, strokovne nasvete in napotke pri izdelavi diplomskega dela. Na tem mestu bi se rada zahvalila družini za potrpežljivost in podporo v času študija. Posebna zahvala gre tudi Maticu Perovšku, Slavku Žitniku, Metki Runovc in vsem prijateljem, ki so me med študijem spodbujali in podpirali.

Kazalo

Seznam uporabljenih kratic in simbolov

2D (*angl.: Two Dimensional*) – dvo-dimenzionalen

3D (*angl.: Three Dimensional*) – tri-dimenzionalen

API (*angl.: Application programming interface*) – aplikacijski programski vmesnik

ERGM (*angl.: Exponential family Random Graph Models*) – eksponentna družina modelov naključnih grafov

GML (*angl.: Graph Modeling Language*) – jezik grafičnega modeliranja

GPL (*angl.: General Public License*) – splošno dovoljenje

GUI (*angl.: Graphical User Interface*) – grafični uporabniški vmesnik

IDE (*angl.: Integrated Development Environment*) – integrirano razvojno okolje

SNA (*angl.: Social network analysis*) – analiza družbenih omrežij

Povzetek

Analiza omrežij pomeni raziskovanje podatkov, ki jih lahko predstavimo z omrežji, npr. računalniški, družbeni, biološki sistemi. V zadnjih letih sta rast svetovnega spleta in večja dostopnost računalnikov omogočila zbiranje in analizo omrežij v velikem obsegu.

Zaradi povečevanja količine in kompleksnosti se velike množice podatkov vse težje učinkovito proučujejo. Podatke, ki so med seboj povezani, lahko predstavimo in vizualiziramo s pomočjo omrežij. Glavni namen vizualizacij je učinkovito posredovati in predstaviti podatke s pomočjo vizualne percepcije ter predvsem olajšati raziskovanje podatkov oziroma omrežja. Veliko programskih orodij pokriva tako področje analize kot tudi vizualizacije, bolj specializirana orodja pa se osredotočajo le na eno izmed teh nalog. Nekatera orodja so razvita posebej za specifično domeno (npr. družbena omrežja).

Gephi, NetworkX, igraph, JUNG in Prefuse so najbolj razširjene knjižnice za analizo in vizualizacijo različnih modelov omrežij. Diplomska naloga obsega pregled in primerjavo funkcij in skalabilnosti teh paketov. Na izbrani zbirki omrežij so knjižnice uporabljene in ocenjene glede na to, kako omogočajo ustvarjanje grafa, spreminjanje lastnosti vozlišč in povezav ter vizualizacijo grafov. Rezultati kažejo, da se ti programski paketi med seboj bolj dopolnjujejo, kot tekmujejo in da so glavne razlike med njimi v izbiri programskega jezika in načinu vizualizacije omrežij.

Ključne besede:

Analiza omrežij, vizualizacija omrežij, Gephi, NetworkX, igraph, JUNG, Prefuse

Abstract

Network analysis is the exploration of data that can be presented with networks, such as computer, social or biological systems. The growth of the Internet and the wide availability of computers have made it possible to gather and analyze network data on a large scale.

Effective examination of large datasets is getting more difficult due to their increase in quantitiy and complexitiy. Datasets that are interconnected can be represented and visualized in with networks. The main purpose of visualization is to effectively present data with the help of visual perception, and in particular to facilitate the exploration of data or networks. There is a set of programming tools that covers the area of both, network analysis and network visualization. Other, more specialized tools, focus only on one of these tasks. Furthermore, there are tools that are developed only for a specific domain (e.g. social networks).

Gephi, NetworkX, igraph, JUNG and Prefuse are libraries that are commonly used for network analysis and visualization. This diploma thesis includes review and comparison of the function coverage and scalability of the chosen software packages. Four datasets are used for performance evaluation. We are particularly interested in graph generation, modification of nodes and edges properties and graph visualization. Results show that these five software packages are complementary rather than competitive. The main differences between them are the choice of programming language and the chosen type of network visualization.

Key words:

Network analysis, network visualization, Gephi, NetworkX, igraph, JUNG, Prefuse

Poglavje 1

Uvod

Hitra rast računalništva in še posebej svetovnega spleta je pripeljala do velikega nabora zbirk podatkov, ki vsebujejo velike količine informacij. Večino teh podatkov je možno predstaviti z omrežji. Primeri vključujejo svetovni splet, družbena omrežja (povezave med prijatelji, sorodniki, sodelavci), organizacijske mreže in mreže poslovnih odnosov med podjetji, nevronske mreže, prehranjevalne verige, omrežja citiranj, omrežja sodelovanj.

Graf je množica objektov, ki jih imenujemo vozlišča, povezava je pa tista, ki združi dve vozlišči med seboj. Omrežje je matematični graf, pri čemer imamo o vozliščih in povezavah dodatno znanje (atributi, lastnosti). Proučevanje grafov je eden bistvenih temeljev diskretne matematike. Eulerjeva rešitev problema Koenigsbergovega mostu iz leta 1735 je pogosto citirana kot prvi dokaz v teoriji omrežij. V dvajsetem stoletju se je teorija grafov razvila v pomembno področje raziskovanja [1].

Omrežja imajo ključni pomen v današnji sodobni družbi, zelo pomembno je razumevanje njihovega delovanja. Velike sisteme podatkov lahko zapišemo v obliki omrežja, z vizualizacijo pa je mogoče izboljšati njihovo razumevanje. Glavni namen vizualizacije je učinkovito posredovanje in predstavitev podatkov s pomočjo vizualne percepcije ter predvsem olajšati raziskovanje podatkov oziroma omrežja. Za analizo in prikazovanje omrežij obstaja veliko orodij - specializirana se osredotočajo samo na analizo ali samo na vizualizacijo, nekatera vključujejo oboje. Vsako omrežje je potrebno zasnovati tako, da najbolj učinkovito prikaže informacije za nadaljnjo analizo. Orodja in paketi za vizualizacijo omrežij se razlikujejo glede na programski jezik, način uvažanja in branja podatkov, obremenjenost spomina itd. Cilj diplomskega dela je predstavitev in primerjava petih pogosto uporabljenih knjižnic, ki omogočajo analizo in predvsem vizualizacijo omrežij.

Preostanek diplomskega dela je razdeljen na 5 poglavij. V naslednjem poglavju predstavimo, kaj so omrežja in kakšne vrste le-teh obstajajo. Podrobneje razložimo pomen analize in vizualizacije omrežij. Tretje poglavje vsebuje pregled obstoječih programov in paketov. V četrtem poglavju so predstavljeni podatki - kakšni so in kaj prikazujejo. Poglavje 5 vsebuje podroben pregled in primerjavo petih izbranih knjižnic ter diskusijo rezultatov primerjave. Na koncu sledita še zaključek in možnosti nadaljnjega dela.

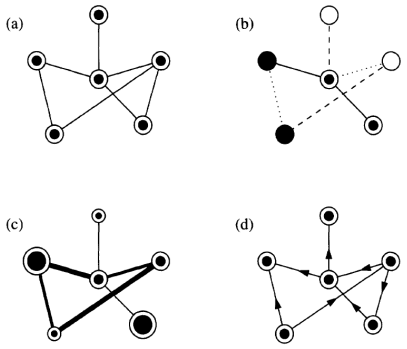
Poglavje 2

Omrežja

2.1 Analiza omrežij

Omrežje predstavlja sistem medsebojno povezanih oseb ali stvari, ki so v neki interakciji (odvisnost, razpoloženje, moč itd.). Množica vozlišč, med seboj združenih s povezavami, je le najpreprostejša vrsta omrežij. Obstaja veliko primerov bolj zapletenih omrežij. Na sliki 2.1 je prikazanih več tipov omrežij: (a) neusmerjeno omrežje z eno vrsto vozlišči in eno vrsto povezavami; (b) omrežje z več diskretnimi vozlišči in povezavami; (c) omrežje z različnimi utežmi vozlišč in povezav; (d) usmerjeno omrežje, kjer ima vsaka povezava svojo smer. V omrežju je lahko več vrst vozlišč in več tipov povezav. Vozlišča in povezave imajo lahko različne lastnosti, številske ali opisne. Če vzamemo primer družbenega omrežja, lahko vozlišča predstavljajo moške ali ženske, ljudi različnih narodnosti, lokacij, starosti, dohodkov. Povezave lahko opisujejo prijateljstva, sovraštva, poslovna poznanstva, geografsko sosedstvo. Povezave imajo lahko uteži, ki pomenijo, na primer v družbenih omrežjih, kako dobro se dve osebi poznata. Povezave so lahko so usmerjene ali neusmerjene. Posledično so omrežja *usmerjena* ali *neusmerjena*. Omrežje, ki prikazuje telefonske klice ali elektronska sporočila med posamezniki, je usmerjeno, saj gre vsako sporočilo vedno v eno smer. Usmerjena omrežja so lahko *ciklična*, kar pomeni, da vsebujejo zaprte zanke povezav, ali *aciklična*, kar pomeni, da takšnih ciklov ne vsebujejo. Omrežja so lahko tudi *delno ciklična*, npr. prehrenjevalne verige [1].

Omrežje je poenostavljena predstavitev, ki zmanjša sistem do abstraktne strukture, pri čemer zajame samo osnove vzorce povezav. Vozlišča in povezave v omrežju imajo lahko oznake z dodatno informacijo, kot so ime, moč, starost vozlišča. S tem pridobimo več podrobnosti o sistemu, kljub temu pa se veliko informacij običajno izgubi v procesu abstrakcije celotnega sistema za predsta-



Slika 2.1: Primeri različnih vrst omrežij [1].

vitev z omrežjem. Zagotovo ima to tako svoje slabosti kot tudi prednosti.

V preteklih letih so znanstveniki na različnih področjih razvili obsežen nabor orodij (matematična, računska in statistična) za analizo, modeliranje in razumevanje omrežij. Večina orodij omogoča prikazovanje preprostega omrežja in po ustreznem računanju hiter vpogled v same lastnosti omrežja, na primer identifikacija najbolj povezanih vozlišč, dolžina poti med vozlišči. Druga orodja omogočajo gradnjo in analizo omrežja ter tako naredijo matematično napoved o procesih, ki potekajo v omrežjih (npr. potek prometa na internetu ali način širjenja bolezni v neki skupnosti). Ker procesirajo omrežja v njihovi abstraktni obliki, lahko taka orodja uporabimo pri večini sistemov, predstavljenih z omrežji. Zagotovo pa niso vsa orodja enako uspešna in učinkovita. Od sistema je odvisno, kakšne meritve in izračune uporabimo pri analizi. Omrežja so zato učinkovito sredstvo za predstavitev vzorcev povezav ali interakcij med deli sistema [2].

Analiza omrežij je zelo uporabna na številnih področjih: v primerih ekonomskih ali finančnih analiz, pri razvoju organizacije, v zdravstveni informatiki. Analiza teroristične mreže lahko pomaga pri razumevanju vlog in relacijskih odnosov med teroristi, njihovimi organizacijami, s tem povezanimi dogodki in nudi druge informacije, ki lahko razkrijejo morebitne šibke točke in odkrijejo morebitne napade. Z analizo družbenega omrežja v neki organizaciji lahko prepoznamo organizacijske vloge posameznikov, zagotovimo poslovne koristi, vključno z ugotavljanjem in ohranjanjem znanja in povezav, izboljšanjem inovacij in produktivnosti, vodenjem z boljšimi odločitvami o organizacijskih spremembah in o oblikovanju ključnih vlog znanja, vpogledom v izzive prenosa znanja in integracijo prihodnjih prestrukturiranj.

2.2 Vizualizacija omrežij

Vizualizacija mora zagotoviti poenostavljen abstraktni pogled celotnega omrežja in hkrati prikazati podrobne informacije določenega vozlišča. Prav tako mora omogočiti analitikom preiskovanje, s čemer pridobijo boljše razumevanje podatkov. Z novimi spoznanji in odkritiji lahko analitiki ovrednotijo in izboljšajo avtomatsko analizo kot tudi vizualizacijo procesov [3].

Obstajajo različne vrste postavitve, ki jih lahko uporabimo pri vizualizaciji omrežij. Najbolj znana strategija je postavitev usmerjenih sil (angl. force-directed layout). Osnovna ideja postavitve je obravnavati graf kot fizikalni model, kjer so vozlišča jekleni obroči, povezave pa vzmeti, pritrjene na obroče. Cilj algoritmov risanja po načinu usmerjenih sil je postaviti vozlišča v 2D (angl. two dimensional) ali 3D (angl. three dimensional) prostoru tako, da so povezave več ali manj istih dolžin in da se čim manj križajo med seboj. Takšen učinek dosežejo z dodelitvijo sil množici povezav in vozlišč na podlagi njihovih relativnih položajev. Z uporabo sil potem simulirajo premikanje povezav in vozlišč za doseganje čim lepšega prikaza.

Fruchterman in Rheingold sta predlagala algoritem, ki uporablja preprost hevristični pristop postavitve usmerjenih sil, ki deluje presenetljivo dobro v praksi. Osnovna ideja je računanje privlačnih in odbojnih sil (angl. attractive and repulsive forces) vsakega vozlišča neodvisno in posodabljati vsa vozlišča iterativno. Sile prikazujejo nekoliko drugačno obliko. Privlačna sila je:

$$f_a(x) = \frac{x^2}{k} \tag{2.1}$$

kjer je k izbran kot:

$$k = \sqrt{\frac{area}{|V|}} \tag{2.2}$$

in odbojna sila je:

$$f_r(x) = \frac{k^2}{x} \tag{2.3}$$

Poleg tega je največji premik vsakega vozlišča v vsaki iteraciji omejen s konstanto, ki se nekoliko zmanjša po vsaki iteraciji.

Drugi zelo znani algoritem, ki deluje po principu usmerjenih sil, se imenuje Kamada-Kawai. Ta algoritem prav tako temelji na ideji o uravnoteženem vzmetnem sistemu in minimizaciji energije. Za razliko od algoritma Fruchterman-Rheingold poskuša uporabiti odvod enačb sil za doseganje hitrejše kon-

vergence. Pri postavitvi Kamada-Kawai je globalna energija definirana kot:

$$E = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{1}{2} k_{i,j} \left(|p_i - p_j| - l_{ij} \right)^2 \tag{2.4}$$

kjer je p_k položaj vozlišča k in $l_{ij} = c \cdot d_{ij}$ je sorazmerna topološki razdalji d_{ij} vozlišč i in j. Postavitev Kamada-Kawai zagotavlja hevristiko, v kateri se premakne samo eno vozlišče naenkrat.

Postavitev usmerjenih sil je zelo uporabna zaradi dobrih hevrističnih metod, ki najdejo primerne prikaze splošnih grafov. Ta vrsta risanja grafov izpostavi celo večino simetrij v danih grafih. Ima sicer tudi slabosti, saj so te metode še zmeraj računsko zahtevne. Ni zagotovila, da bo prikaz optimiziran, saj križanje povezav pogosto nastane tudi v ravninskih grafih. Zato so te tehnike praktično uporabne samo za idealne grafe, kar pomeni, da se razširitve in oblike vozlišč ali položaji oznak ne upoštevajo. Kljub temu so metode usmerjenih sil med najbolj priljubljenimi pristopi [4].

Obstajajo še druge vrste postavitev, ki so vezane na obliko prikaza: metode pravokotne postavitve (angl. orthogonal layout methods), kjer se povezave razporedijo vzporedno glede na koordinatni sistem prikaza grafa; algoritmi drevesne postavitve (angl. tree layout algorithms), ki so namenjeni grafom drevesnih oblik; metode risanja slojne postavitve (angl. layered graph drawing), ki so najbolj primerne za usmerjene aciklične grafe; ločni diagrami (angl. arc diagrams), kjer so vozlišča postavljena v ravno črto, povezave med njimi pa potekajo nad ali pod črto v obliki polkrogov; krožna postavitev (angl. circular layout), ki postavi vozlišča na krožnico. Na voljo je še veliko drugih načinov postavitve grafov [5].

Poglavje 3

Pregled paketov in programov

V zadnjih desetletjih so raziskovalci razvili različne tehnike za analizo in vizualizacijo omrežij. Priljubljenost družbenih omrežij (angl. Social Network Analysis, SNA) je povzročila razmah orodij, ki so posebej namenjena njihovi analizi.

Omrežja so lahko sestavljena iz projektnih skupin, družin, učilnic, športnih ekip, zakonodaj, držav, prenašalcev bolezni, članstev na spletnih straneh (npr. Twitter in Facebook). Vozlišča so lahko povezana z neposrednimi ali posrednimi povezavami, ki temeljijo na skupnih lastnostih, skupni udeležbi na prireditvah ali skupnih zvezah. Lastnosti omrežij so lahko na ravni posameznih vozlišč, skupin vozlišč ali celotnega omrežja. Primeri lastnosti vozlišč so vmesnost (angl. betweenness) in središčnost (angl. centrality) ali atributi, kot so starost, spol, dohodek. Programi generirajo te lastnosti iz neobdelanih podatkov, ki so lahko v različnih oblikah: seznam povezav, seznam sosednosti ali matrika sosednosti. V formatu seznamov povezav so povezave med vozlišči naštete v vsaki vrstici datoteke. V vsaki vrstici sta lahko dve ali več povezav, ločene med seboj s piko, podpičjem ali presledkom. V formatu seznamov sosednosti je možno napisati vse povezave vozlišč v eni vrstici. Na začetku vsake vrstice zapišemo izvorno vozlišče, ki mu sledijo vsi njegovi sosedi oziroma ponorna vozlišča. V matriki sosednosti so vsa vozlišča našteta v prvi vrstici in v prvem stolpcu. Vsa ostala polja vsebujejo 0 ali 1 (lahko tudi drugo številko, ki predstavlja utež povezav). Kadar je številka večja od 0, pomeni, da je med tema dvema vozliščema povezava.

Programi za analizo omrežij so na splošno sestavljeni iz paketov, ki temeljijo na grafičnih uporabniških vmesnikih (angl. graphical user interfaces, GUIs), ali iz paketov, ki so zgrajeni za skriptne/programske jezike. GUI paketi so enostavni za uporabo, medtem ko so skriptna orodja močnejša in omogočajo

razširitve. Primeri splošno uporabnih in dobro dokumentiranih GUI paketov so: *NetMiner*, *UCINet*, *Pajek*, *GUESS*, **ORA* in *Cytoscape*. Pogosto uporabljena in dobro dokumentirana skriptna orodja za analizo omrežij vključujejo: *NetMiner* s skriptnim jezikom Python, *statnet* nabor paketov s statističnim programskim jezikom R, *igraph*, ki ima pakete za R in Python, *NetworkX* knjižnica za Python in *SNAP* paket za analizo velikih omrežij v programskem jeziku C++. Čeprav so težje naučljivi za uporabnike brez računalniškega predznanja, nekatere od teh odprtokodnih paketov še vedno razvijajo in izboljšujejo funkcionalnosti. Na voljo so tudi obsežne dokumentacije in primeri uporabe.

Vizualna predstavitev je pomembna za razumevanje omrežja in prikazovanje rezultatov analize. Vizualizacija pogosto omogoča tudi kvalitetno interpretacijo samih podatkov. Za lepšo vizualizacijo imajo orodja za analizo omrežij možnost spreminjanja postavitve, barv, velikosti in drugih lastnosti vozlišč in povezav. Vsa zgoraj našteta orodja omogočajo vizualizacijo; *NetMiner*, *igraph*, *Cytoscape*, *NetworkX* ustvarijo najbolj kakovosten grafični prikaz omrežij [6].

3.1 NetMiner

NetMiner je programska aplikacija za raziskovalno analizo in vizualizacijo omrežja. Orodje omogoča raziskovalcem raziskovanje omrežja vizualno in interaktivno, prav tako jim pomaga odkriti osnovne vzorce in strukture omrežja. *NetMiner* vsebuje obdelavo podatkov, analizo omrežij, vizualizacijo omrežja, diagrame, statistike in temelji na skriptnem programskem jeziku Python. *NetMiner* je bil izdan v letu 2001 kot programska oprema za poslovno analizo družbenih omrežij. Obstajajo različne licence, ne samo za poslovno uporabo, ampak tudi za akademske potrebe. Trenutna različica programa je *NetMiner 4* za Microsoft Windows (2000 ali novejše verzije). *NetMiner 4* je na voljo študentom in učiteljem brezplačno za obdobje treh mesecev.

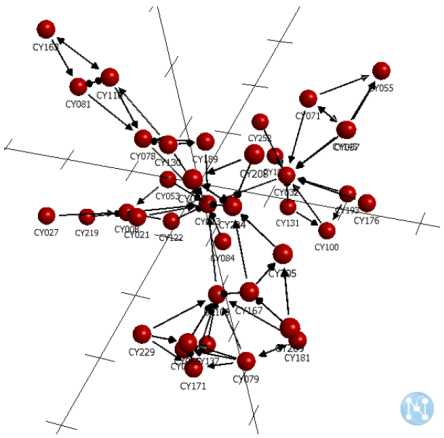
NetMiner lahko analizira velika omrežja sestavljena iz največ milijon vozlišč. Vsebuje 100 modulov za analizo/vizualizacijo omrežij, ter tudi splošne statistične module za analizo in prikazovanje z razpredelnico.

NetMiner omogoča tako grafični kot tudi skriptni način uporabe. S tem zagotavlja najboljše iz obeh svetov - enostavna uporaba pri grafičnem načinu in razširljivost pri skriptnem. Skriptni *NetMiner* je grafično zasnovan, kar pomeni, da je izvedba v skriptnem načinu popolnoma enaka izvedbi v grafičnem načinu, ne samo v smislu rezultatov, ampak tudi v samem procesu. Z uporabo skript lahko uporabnik obravnava procese bolj natančno.

Glavne značilnosti programa *NetMiner* vključujejo:

- analizo velikih omrežij;
- analizo Kaj-če (angl. What-if) omrežij;
- obsežne meritve in modele omrežja;
- upravljanje delovnega toku produktivnosti;
- upravljanje s podatki;
- interaktivno analizo vizualizacije;
- prijazno do uporabnika okolje;
- vgrajene funkcije/module za statistično analizo in prikazovanje z razpredelnicami.

NetMiner 4 je prvi SNA program, ki omogoča vizualizacijo omrežja v 3D obliki. To ne pomeni samo prikaz vozlišč in povezav kot 3D objektov, ampak je celotno omrežje predstavljeno v treh dimenzijah (slika 3.1). Prav tako lahko uporabnik po želji spreminja kot pogleda na sliko. Uporabniki lahko tako omrežje raziskujejo bolj enostavno in zelo podrobno.



Slika 3.1: Primer 3D prikaza omrežja [7].

NetMiner se uporablja za splošna raziskovanja ter poučevalne in strokovne analize v družbenih omrežjih. Poleg tega je učinkovit tudi na različnih poslovnih področjih, kjer lahko strukturni faktorji omrežja kritično vplivajo na delo: npr. organizacijska, finančna, spletna, kriminalna, telekomunikacijska, distribucijska in prometna omrežja [8].

3.2 UCINET

UCINET je orodje, ki se uporablja za upravljanje in analizo podatkov. Program vključuje veliko različnih možnosti upravljanja z omrežji. Zahtevan operacijski sistem za namestitev je Windows. Program so razvili Lin Freeman, Martin Everett in Steve Borgatti pri podjetju Analytic Technologies in je plačljiv. *UCINET* zna prebrati in pisati podatkovne datoteke družbenih omrežij. Orodje se največ uporablja v akademskem raziskovalnem svetu. Format datotek *UCINET* se lahko uporablja pri drugih platformah za analizo in vizualizacijo omrežij. *UCINET* uporabljajo tudi svetovalci podjetij, ki so razvili prilagojene različice programa za lastne potrebe.

Orodje ima možnost uvoza podatkov iz različnih formatov datotek, vključno z datotekami programa Excel. Združljivo je z mnogimi drugimi platformami za vizualizacijo omrežij. *UCINET* je menijsko usmerjen program. To pomeni, da se predmeti izbirajo iz dostopnega menija. Meniji so lahko vgnezdeni, saj lahko vsebujejo tudi podmenije. Da pridemo do določenih možnosti, je včasih potrebno izbrati več menijev na poti.

Orodje *UCINET* omogoča več različnih tipov analize v smislu uporabe (odkrivanje podskupin, merjenje središčnosti itd.) in izvora (kdo jih je razvil, različni posamezniki različnih matematičnih, metodoloških in vsebinskih vidikov).

Vsi podatki v programu *UCINET* so opisani z matrikami. Razumevanje, kako so omrežja, grafi, hipergrafi, relacije in vse druge entitete analize omrežij prikazani kot matrike, je bistvenega pomena za učinkovito in nemoteno uporabo sistema [9].

3.3 NetDraw

NetDraw je program za vizualizacijo omrežij, ki je bil razvit v podjetju Analytic Technologies, tako kot *UCINET* (poglavje 3.2). Program omogoča grafični prikaz omrežij, vključno z relacijami in atributi. Ima nekatere analitične funkcije, ki se delno prekrivajo s funkcijami programa *UCINET*.

NetDraw vsebuje številne algoritme za različne postavitve vozlišč v 2D prostoru in za izvajanje osnovne analize omrežij. Orodje temelji na neposredni interakciji, kar omogoči uporabnikom upravljanje in spreminjanje grafov. To poenostavi njihovo analizo in interpretacijo podatkov.

Program *NetDraw* bere sistemske datoteke tipa *UCINET*, tekstovne datoteke UCINET DL in tekstovne datoteke Pajek (vključno z .net, .clu in .vec).

Diagrami se lahko izvozijo v formate EMF, WMF, BMP in JPG. Program omogoča tudi neposredno tiskanje v visoki ločljivosti.

Glavne značilnosti programa *NetDraw* so:

- program je enostaven za namestitev in uporabo;
- ponuja številne možnosti, ki omogočajo uporabnikom prikaz dodatnih informacij omrežja (kot so moč povezav, velikost vozlišč, atribute vozlišč itd.);
- orodje je v celoti poenoteno s programom *UCINET*;
- bere vhodne tekstovne datoteke;
- integrira orodje *Pajek* (poglavje 3.4).

Slaba stran programa je, da je na voljo samo za operacijski sistem Windows. Prav tako sistemska dokumentacija še ni v celoti razvita [10].

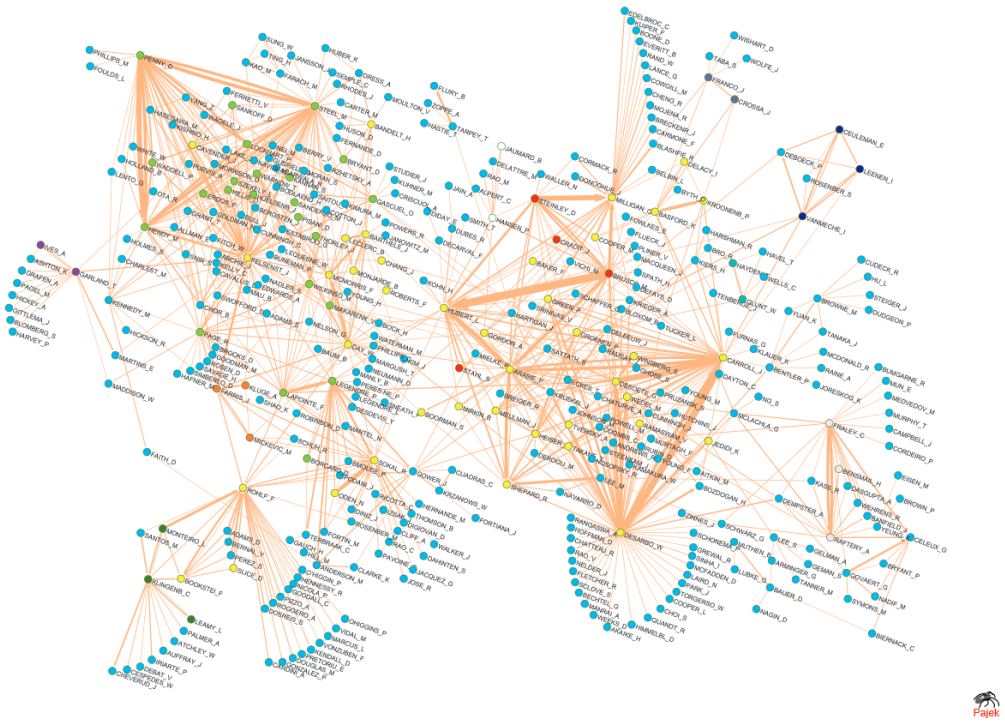
3.4 Pajek

Pajek je program za operacijski sistem Windows za analizo in vizualizacijo velikih omrežij z nekaj tisoč ali celo milijoni vozlišč. Najnovejša različica programa je na voljo brezplačno za nekomercialno uporabo [11].

Program *Pajek* sta začela razvijati Vladimir Batagelj in Andrej Mrvar leta 1996. Napisan je v jeziku Delphi (Pascal). Nekatere procedure je prispeval Matjaž Zaveršnik.

Glavna motivacija za razvoj programa *Pajek* je bila ugotovitev, da obstaja več virov velikih omrežij, ki so v strojno berljivi obliki. *Pajek* vsebuje vrsto orodij za analizo in vizualizacijo omrežij, kot so: omrežja sodelovanja, organske molekule v kemiji, omrežja interakcij med proteini, rodoslovna drevesa, omrežja svetovnega spleta, omrežja citiranj, razpršitvena omrežja (AIDS, novice, novosti) itd. Slika 3.2 prikazuje primer omrežja citatov. Med avtorji je mogoče najti znane člane IFCS (angl. Intergovernmental Forum on Chemical Safety) skupnosti.

Zasnova programa *Pajek* temelji na prejšnjih izkušnjah avtorjev, pridobljenih pri raziskavah grafovskih struktur in algoritemskih knjižnic Graph in X-Graph, zbirki programov za analizo in vizualizacijo omrežij STRAN, RelCalc, Draw, Energ, in SGML (angl. Standard Generalized Markup Language) opisni označevalni jezik na osnovi grafa (angl. based graph description markup language) NetML. Glavni cilji pri oblikovanju programa *Pajek* so bili:



Slika 3.2: Omrežje citiranj, v katerem vozlišča predstavljajo avtorje, usmerjene povezave pa kažejo na citate med njihovimi članki. Z uporabo uteži povezav se določijo t.i. otoki (majhna povezana podomrežja) [13].

- podpreti abstrakcijo z (rekurzivno) dekompozicijo velikih omrežij na več majhnih omrežij, ki se lahko dodatno razvija s pomočjo visoko razvitih metod;
- ponuditi uporabniku nekaj močnih orodij za vizualizacijo;
- izvesti selekcijo učinkovitih algoritmov za analizo velikih omrežij.

S programom *Pajek* lahko poiščemo gruče oziroma skupine (kompoment, sosednosti pomembnih vozlišč, jedra itd.) v omrežju; izločimo vozlišča, ki pripadajo isti gruči in jih prikažemo ločeno, po možnosti z delnimi konteksti (podroben lokalen pogled); skrčimo vozlišča in gruče ter prikažemo ralcije med gručami (globalni pogled).

Poleg navadnih (usmerjenih, neusmerjenih, mešanih) omrežij *Pajek* podpira tudi dvodelni način (dvodelna omrežja) - to so omrežja med dvema nepoveza-

nima množicama vozlišč - in začasna omrežja (dinamični grafi) - to so omrežja, ki se spreminjajo skozi čas [12].

3.5 GUESS

GUESS je orodje za preiskovalno analizo in vizualizacijo omrežij. Sistem temelji na programskem jeziku imenovanem Gython (razširitev programskega jezika Python ali natančneje Jython), ki podpira operacije in sintakso za delo z grafi. Interaktivni prevajalnik povezuje besedilo, ki je vneseno v prevajalnik z vizualiziranimi objekti. *GUESS* ponuja tudi čelni del (angl. frontend) vizualizacije, ki podpira izvoz statičnih slik in dinamičnih videov.

GUESS je zasnovan na programskem jeziku Jython/Java, zato je mogoče aplikacije in programe napisati brez veliko programiranja. *GUESS* je napisan pod licenco GPL (angl. General Public License) in omogoča prosto uporabo v številnih odprtokodnih projektih in prispevkih. Program *GUESS* je razvit in vzdrževan s strani Eytana Adarja (prvotno v podjetju Hewlett-Packard). Prejšnja različica sistema Zoomgraph je še vedno na voljo na spletni strani HP [14].

Orodje vsebuje:

- celotno preoblikovanje različice sistema za vizualizacijo grafov Zoomgraph;
- robusten jezik za selekcijo in upravljanje vozlišč in povezav. Jedro jezika Jython (Python in Java) je bilo razširjeno za posebno sintakso programa *GUESS*;
- uporabo *JUNG* (predstavljeno v poglavju 3.17), robustne knjižnice za grafe, kot zaledni del (angl. backend) za predstavitev vozlišč in grafov;
- povečevalni vmesnik za velike grafe. Omogočeno je nemoteno povečevanje in pomanjševanje vozlišč. Poleg tega nova različica počasi pridobiva tudi podporo za upodabljanje (angl. rendering) grafov v knjižnici *Prefuse* (predstavljeno v poglavju 3.18) ali TouchGraph (deluje v omejenem načinu trenutno);
- sistem zasnovan s podatkovno bazo. Vozlišča in povezave imajo značilnosti, ki jih lahko preiskujemo in uporabljamo za nadzorovanje prikazanega;
- zapisovanje različnih vrst formatov (JPG, GIF, PDF, EPS, SVG, SWF);
- različne algoritme za postavitev;

- vmesnik za programski jezik R;
- podporo podgrafov [15].

3.6 *ORA

**ORA* (angl. Organizational Risk Analysis) je dinamično orodje za ocenjevanje in analizo omrežij. Razvil ga je CASOS (angl. center for Computational Analysis of Social and Organizational System) na Univerzi Carnegie Mellon (angl. Carnegie Mellon University, CMU). Vsebuje metrike za dinamična omrežja, sledljive metrike, procedure za združevanje vozlišč, prepoznavanje lokalnih vzorcev, primerjanje in razlikovanje omrežij, skupin in posameznikov iz vidika dinamičnega omrežja. **ORA* lahko upravlja z omrežji v več načinih in na več nivojih. Program dobro deluje pri analizi velikih omrežij. Del tega orodja so algoritemske in statistične procedure za primerjavo omrežij.

**ORA* lahko prepozna ključne člane, skupine in ranljiva mesta omrežja med različnimi časovnimi točkami in na podlagi tega lahko nadaljuje z nadaljnjo analizo omrežja. V primerjavi z ostalimi orodji za analizo socialnih omrežij, **ORA* podpira več formatov vhodnih podatkov in omogoča takojšen prikaz dinamičnih sprememb v omrežju.

Veliko meritev kritičnosti, ki temeljijo na teoriji omrežij, socialni psihologiji, operacijskih raziskavah in teoriji upravljanja, je bilo razvito na CMU. Tako, kot se lahko algoritmi kritične poti uporabijo pri iskanju teh kritičnih nalog iz vidika projektnega upravljanja, lahko algoritmi programa **ORA* najdejo ljudi, vrste znanja, sposobnosti in naloge, ki so kritični iz vidika produktivnega dela in informacijske varnosti [16].

**ORA* se lahko uporabi v podjetju za ugotavljanje nevarnosti pri rizičnih posameznikov ali organizacij. Takšne nevarnosti vsebujejo, npr. nagnjenost k skupinskem razmišljanju, spregled informacij, komunikacijske ovire in kritične zaposlene. Ocenjuje tudi morebitne nevarnosti v družbenih omrežjih, omrežij znanja, virov in nalog. Program **ORA* je bil uporabljen za ocenjevanje nevarnosti v različnih organizacijskih in državnih okoljih, vključno z NASA, bolnišnicami in vojskami [17].

3.7 Cytoscape

Cytoscape je odprtokodna programska platforma za vizualizacijo interakcijskih molekularnih omrežij in bioloških poti ter povezovanje teh omrežij z označ-

bami, izražanjem genskih profilov in drugimi podatki. Čeprav je bil *Cytoscape* prvotno zasnovan za boiloške raziskave, je sedaj splošna platforma za analizo in vizualizacijo kompleksnih omrežij. Osnovna verzija programa *Cytoscape* zagotavlja osnoven nabor funkcij za integracijo podatkov, analizo in vizualizacijo. Dodatne funkcije so na voljo kot applikacije. Aplikacije ponujajo analizo omrežij in molekularne profilirane analize, nove postavitve, dodatno podporo datotečnih formatov in povezavo z bazami podatkov. Aplikacije lahko razvija vsak uporabnik odprtokodnega *Cytoscape API* (angl. application programming interface), ki temelji na programskem jeziku Java. Večina aplikacij je na voljo brezplačno v trgovini *Cytoscape App*.

Obstajajo tri različice *Cytoscape*: *Cytoscape 2.x* (stabilna produkcijska verzija), *Cytoscape 3.x* (nova verzija z modularno arhitekturo; pričakuje se, da bo nadomestila 2.x) in *cytoscape.js* (JavaScript knjižnica za vizualizacijo omrežij, ki pa ni samostojna spletna aplikacija).

Cytoscape podpira veliko standardnih formatov datotek, vključno z: SIF (angl. Simple Interaction Format), GML (angl. Graph Modeling Language) , XGMML, BioPAX, PSI-MI, GraphML, KGML (KEGG XML), SBML (angl. Systems Biology Markup Language), OBO (angl. Open Biological and Biomedical Ontologies) in GO (angl. Gene Ontology). Omrežja lahko izvozimo v različnih formatih: PDF, PS, SVG, PNG, JPEG in BMP. Postavitev omrežij je v 2D obliki. Na voljo je veliko različnih postavitev: krožna, drevesna, usmerjenih sil, utežena in yFiles organična. Orodje omogoča povečevanje in pomanjševanje omrežij. Prav tako lahko filtriramo omrežja z izbiro podmnožic vozlišč in/ali interakcij, ki temelijo na določenih podatkih. Zatem lahko ustvarimo novo omrežje iz rezultata filtriranja. Program *Cytoscape* je sposoben najti gruče v omrežju. Odvisno od vrste omrežja, gruče pomenijo različne stvari (npr. gruče v proteinskem omrežju podobnosti predstavljajo beljakovinske družine). Druga prednost programa *Cytoscape* je to, da so podatki v datotekah lahko v različnih jezikih, kar pa ne vpliva na vizualizacijo [18].

3.8 Statnet

Statnet je zbirka programskih paketov za statistično analizo omrežij. Analitično ogrodje temelji na eksponentni družini modelov naključnih grafov (angl. Exponential family Random Graph Models, ERGM). Sestavni deli paketa zagotavljajo obsežen okvir za modeliranje omrežja, ki temeljijo na ERGM, vključno z orodji za ocenjevanje modelov, vrednotenje modelov, simulacijo omrežij in vizualizacijo omrežij. To obsežno funkcionalnost poganja osrednji

algoritem Monte Carlo za Markovske verige (angl. Markov chain Monte Carlo). Ta algoritem lahko enostavno upravlja z omrežji, ki imajo več tisoč vozlišč.

Zbrika *statnet* se lahko uporablja za analizo različnih vrst omrežij iz različnih vsebinskih področij, vključno z usmerjenimi ali neusmerjenimi in enodelnimi ali dvodelnimi omrežji. Obstajajo nekatere funkcionalnosti za upravljanje dinamičnih omrežij in omrežji z manjkajočimi podatki, kar je še vedno v razvoju.

Statnet ima drugačen cilj kot ga ima na primer *Pajek* (poglavje 3.4). Poudarek je na statističnem modeliranju podatkov. Paketi programa *statnet* so napisani v kombinaciji odprtokodnega statističnega jezika R s programskim jezikom C in se kličejo v ukazni vrstici jezika R. Ker tečejo v paketu R, je na voljo tudi celotna funkcionalnost samega jezika. Posamezni sestavni deli paketa *statnet* so navedeni spodaj.

3.8.1 Standardni paketi

- *ergm* je zbirka funkcij za prilagajanje, simulacijo, izris in določanje ERGM grafov. Osnovne funkcije paketa *ergm* so: *ergm*, funkcija za prilagajanje eksponentnih družin modelov naključnih grafov, pri katerih je verjetnost omrežja odvisna od vektorja statistike omrežja, določenega s strani uporabnika; *simulate*, funkcija za simulacijo naključnih omrežij z uporabo ERGM; *gof*, funkcija, ki oceni ustreznost prilagajanja podatkov z ERGM.

- *network* je paket za ustvarjanje, shranjevanje, spreminjanje in izris podatkov v obliki omrežja. Objektni razred *network* definiran v paketu *network* prikazuje nabor relacijskih podatkovnih tipov in podpira poljubne atribute vozlišč/povezav/omrežja. Podatke, shranjene kot objekti, lahko potem analiziramo z uporabo paketov zbirke *statnet*.

3.8.2 Opcijski paketi

Opcijski paketi *sna*, *degreenet*, *latentnet* in *networksis* so na voljo na CRAN omrežju (angl. Comprehensive R Archive Network).

- *sna* je množica orodij za tradicionalno analizo družbenih omrežij;

- *degreenet* je paket, ki je bil razvit za razporeditev stopenj v omrežjih;

- *latentnet* je paket za prilagajanje in ocenjevanje latentnega položaja (angl. latent position) in modelov gruč v statističnih omrežjih;

- *networksis* je paket za simulacijo dvodelnih grafov s fiksnimi robovi. Pri tem uporablja zaporedno vzorčenje pomembnosti (angl. sequential importance sampling) [19].

3.9 SNAP

SNAP (angl. Stanford Network Analysis Platform) je splošno namenski, visoko zmogljiv sistem za analizo, upravljanje in spreminjanje velikih omrežij. Jedro knjižnice je napisano v jeziku C++. Knjižnica je optimizirana za maksimalno učinkovitost in jedrnato predstavitev grafov. Omogoča predstavitev obsežnih omrežj z več sto milijonov vozlišči in milijardami povezavami. *SNAP* učinkovito upravlja z velikimi grafi, računa strukturne lastnosti, ustvarja običajne in naključne grafe in ima možnost dodajanja in uporabe atributov vozlišč in povezav. Poleg upravljanja z velikimi grafi, je prednost knjižnice *SNAP* dinamično spreminjanje vozlišč, povezav in atributov omrežja v času računanja.

SNAP deluje na operacijskih sistemih Windows z naloženim programskim paketom Visual Studio ali Cygwin, Mac OS X in Linux. Knjižnica je v veliki meri samostojna in ima minimalne zahteve odvisnosti od drugih paketov.

Snap.py je vmesnik za knjižnico *SNAP* v programskem jeziku Python. *Snap.py* zagotavlja prednost zmogljivosti knjižnice *SNAP* v kombinaciji s fleksibilnostjo programskega jezika Python. *Snap.py* zahteva nameščeno različico 2.7.x jezika Python. Podprte so različice za operacijski sistemi Mac OS, Linux in 64 bitni Windows.

SNAP je prvotno razvil Jure Leskovec v okviru doktorskega študija. Prva različica je bila na voljo leta 2009. *SNAP* uporablja knjižnico za splošno rabo *GLib*, tipa STL (angl. Standard Template Library), ki je bila razvita na Inštitutu "Jožef Stefan". Knjižnici *SNAP* in *GLib* sta v aktivnem razvoju in se uporabljata v številnih akademskih in industrijskih projektih [20].

3.10 Graphviz

Graphviz je odprtokodni program za vizualizacijo grafov, katerega razvoj je pričel raziskovalni laboratorij AT&T za risanje grafov. Na voljo je tudi kot knjižnica za uporabo v drugih programskih aplikacijah. *Graphviz* je brezplačni program pod licenco Eclipse Public.

Graphviz omogoča več osnovnih načinov postavitve. Te postavitve se lahko uporabijo preko vmesnika C knjižnice, preko orodij z ukaznimi vrsticami, preko

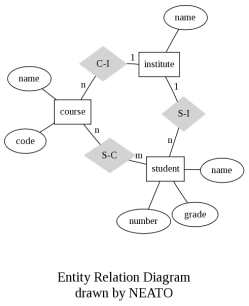
GUI ali spletnih brskalnikov. Prav tako vsebuje spletne in interaktivne grafične vmesnike, pomožna orodja, knjižnice in doseganje jezikov (angl. language binding). Obstajajo nekaj zunanjih projektov in komercialnih orodij, ki vključujejo *Graphviz*. Načini postavitve jemljejo opise grafov v enostavnem tekstovnem jeziku in naredijo diagrame v uporabnih formatih, kot so slike in SVG (angl. Scalable Vector Graphics) za spletne strani; PDF in Postscript za vključitev v druge dokumente; ali prikaz interaktivnega grafa v brskalniku.

Graphviz ima številne uporabne funkcije za posebne diagrame, kot so omogočanje barvanja, različne pisave, pregledne postavitve vozlišč, sloge črt, hiperpovezave in prilagojene oblike. V praksi so grafi generirani iz zunanjih virov podatkov, lahko pa jih ustvarimo in urejamo ročno: bodisi kot neobdelane datoteke z besedilom, bodisi v grafičnem urejevalniku.

Program *Graphviz* je sestavljen iz jezika opisovanja grafov (angl. graph description language) imenovan jezik DOT in iz nabora orodij, ki lahko ustvarjajo in/ali obdeljujejo DOT datoteke:

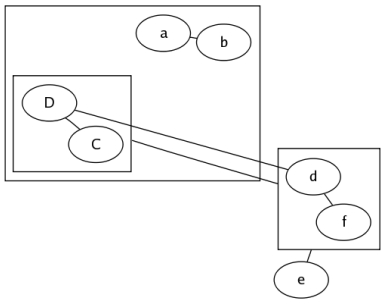
- *dot* - hierarhično ali večnivojsko risanje usmerjenih grafov. Orodje, ki se privzeto uporablja za usmerjene povezave. Orodje *dot* usmeri povezave v isto smer (od zgoraj navzdol ali od leve proti desni) in nato poskuša preprečiti križanje povezav in zmanjša dolžine povezav.

- *neato* - postavitev "sping modela" (slika 3.3). Privzeto orodje za uporabo, če graf ni prevelik (okoli 100 vozlišč) in če ne vemo nič drugega o grafu. Orodje *neato* uporablja naključnost, tako da vsakič naredi drugačno postavitev.



Slika 3.3: Entitetni relacijski model [21].

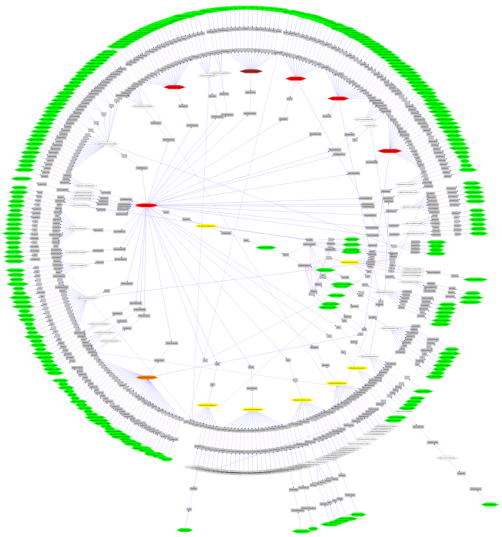
- *fdp* - postavitev "spring modela" je podobna postavitvi *neato*. Orodje *fdp* uporablja hevristiko Fruchterman-Reingold in ima možnost obravnave večjih grafov in gruč neusmerjenih grafov (slika 3.4).



Slika 3.4: Orodje *fdp* podpira povezave tipa vozlišča - gruče in gruče - gruče [21].

- *sfdp* - večnivojska različica orodja *fdp* za predstavitev velikih grafov.

- *twopi* - radialna postavitev. Vozlišča se postavijo v koncentrične kroge v odvisnosti od njihove oddaljenosti od določenega korenskega vozlišča (slika 3.5).



Slika 3.5: Omrežje, ki vsebuje 300 mest v več kot 40 državah [21].

- *circo* - krožna postavitev. Orodje je primerno za določene diagrame večcikličnih struktur, kot so nekatera telekomunikacijska omrežja [21].

3.11 Tulip

Tulip je ogrodje za informacijsko vizualizacijo, namenjeno analizi in vizualizaciji relacijskih podatkov. Ogrodje *Tulip* je napisano v programskem jeziku C++ in omogoča razvoj algoritmov, vizualno kodiranje, interakcijske tehnike, podatkovne modele in vizualizacije za določena področja. Eden od ciljev ogrodja je omogočanje ponovne uporabe komponent. Ta možnost oblikuje ogrodje kot bolj učinkovito za raziskovanje prototipov in razvoj končnih aplikacij. Program *Tulip* je sestavljen iz orodij za razvrščanje v gruče, risanje grafov in barvanje metrik v okviru informacijske vizualizacije [22].

Tulip ima naslednje funkcionalnosti:

- vizualizacija in navigacija v dveh ali treh dimenzijah;
- podpora velikih grafov;
- podpora spreminjanja grafov;
- upravljanje gruč;
- upravljanje neomejenega števila skupnih lastnosti posameznih grafov;
- mehanizem za ocenjevanje notranjih lastnosti;
- ponovna uporaba komponent, ne da bi bilo potrebno ponovno prevajanje programa;
- brezplačen za uporabo in odprtokoden program.

Pri vizualizaciji grafov ima *Tulip* različne možnosti, kot so spreminjanje postavitve, uporabo različnih metrik pri elementih grafa, izbiro vozlišča, povezav ali podgrafov, spreminjaje barve, velikosti ali oblike, prikazovanje in dodajanje oznak. Program *Tulip* zagotavlja grafični vmesnik za vizualizacijo in urejanje podatkovnih struktur in lastnosti, ki so naštete zgoraj. Program je sestavljen iz dveh vrst oken. Aplikacijsko okno poskrbi za osnovna programska orodja. Drugo okno je okno za nadgrafe (angl. super-graph window), ki omogoča izvajanje operacij specifičnih za nadgrafe (v teoriji grafov: če je A podgraf od B-ja, potem je B nadgraf A-ja). Aktivnih je lahko več oken za nadgrafe hkrati [23].

Tulip Python je množica modulov, ki v programskem jeziku Python omogočajo skoraj vse funkcionalnosti, ki jih ima na voljo Tulip C++ API. Glavne značilnosti, ki jih ti moduli ponujajo so: ustvarjanje, upravljanje in spreminjanje grafov; shranjevanje podatkov elementov grafa (float, integer, boolean, barva,

velikost, koordinate, seznami itd.); aplikacija algoritmov različnih vrst grafov (postavitve, metrike, razvrščanje v gruče itd.); možnost pisanja novih vtičev (angl. plugins) v programskem jeziku Python. Moduli se lahko uporabljajo znotraj Tulip GUI pri zaganjanju skript na trenutni vizualizaciji. Od različice 3.6 programa *Tulip* naprej se moduli lahko uporabijo izven *Tulip* v klasičnem prevajalniku za programski jezik Python [24].

3.12 SocNetV

SocNetV (angl. Social Networks Visualizer) je prilagodljivo in do uporabnika prijazno orodje za analizo in vizualizacijo družbenih omrežij. Omogoča ustvarjanje in nalaganje omrežij v različnih formatih ter njihovo spreminjanje za specifične potrebe. Orodje *SocNetV* ponuja tudi vgrajen spletni preiskovalnik (angl. web crawler), ki omogoča samodejno ustvarjanje omrežij vseh povezav, ki so bile najdene na začetnem URL (angl. uniform resource locator) naslovu. Druge funkcije programa so: interakcija in izbira skupin; nalaganje omrežja in shranjevanje aktivnega omrežja; pregled matrike sosednosti.

Aplikacija lahko izračuna osnovne lastnosti omrežja, kot so gostota, premer in razdalje (dolžine najkrajših poti). Izračuna lahko tudi bolj napredne strukturne statistike, kot so vmesnost in središčnost vozlišč in omrežij, koeficient razvrščanja v gruče itd.

Podprti so različni algoritmi postavitve (npr. spring, radialna in večplastna glede na središčnost vozlišč (angl. node centralization)) za smiselne vizualizacije omrežij. Poleg tega orodje omogoča ustvarjanje naključnih omrežij.

SocNetV je še vedno v razvoju. Razvit je v programskem jeziku C in aplikacijskem ogrodju Qt. Program *SocNetV* je brezplačen ter odprtokoden pod licenco GPL 3. Požene se lahko v vseh operacijskih sistemih: Linux, Windows ali OS X (za Windows in OS X je potrebno imeti nameščene knjižnice Qt4) [25].

3.13 Visone

Visone je dolgoročni raziskovalni projekt, v katerem se razvijajo modeli in algoritmi za vizualizacijo in analizo družbenih omrežij. Pomemben del projekta *Visone* je načrtovanje in implementacija programskega orodja, ki je namenjena za raziskave in poučevanje pri analizi družbenih omrežij. Zasnovan je tako, da lahko strokovnjaki in začetniki z lahkoto in natančnostjo uporabljajo inovativne in napredne vizualne metode. Poganja se s pomočjo programskega

jezika Java in se lahko naloži ali uporablja preko spleta. Temelji projekta so bili postavljeni na Univerzi v Konstanzu, danes pa ga razvijajo še na drugih univerzah.

Glavne značilnosti programa *Visone* so:

- interaktivni GUI, prilagojen za družbena omrežja;
- podpora ralacij;
- na voljo za operacijske sisteme Windows, Linux in MacOS;
- uvoz in izvoz standardnih formatov podatkov;
- izvoz kvalitetnih formatov JPEG, PDF, SVG, Metafile.

Za enostavno uporabo, implementacijo in dokumentacijo *Visone* uporablja enoten model omrežja (označen usmerjeni graf), ki je dovolj splošen za zajem bistvenih značilnosti v večini možnih primerov. Orodje je interaktivno, kar omogoča spremembo podgrafa, nad katerim je bila izvedena določena operacija.

Orodje *Visone* vsebuje mehanizme za upravljanje s poljubnim številom atributov vozlišč in povezav. Atribute lahko prikažemo v okviru vizualizacije grafa ali jih lahko uporabimo za določanje dolžine in moč oznak, kar vpliva tudi na analizo omrežja [26].

3.14 Gephi

Gephi je interaktivna platforma za vizualizacijo in raziskovanje vseh vrst omrežij, dinamičnih in hierarhičnih grafov. Deluje na operacijskih sistemih Windows, Linux in Mac OS X. *Gephi* je odprtokodni in brezplačni program pod GNU GPL licenco. Obstaja tudi različica programa v obliki knjižnice. Obe platformi temeljita na programskem jeziku Java.

Gephi je orodje za raziskovanje in razumevanje grafov. Uporabnik uporablja prikaz, upravlja in spreminja strukture, oblike in barve, da lahko analizira lastnosti grafa. Cilj je pomagati podatkovnim analitikom ustvariti hipoteze, intuitivno odkriti vzorce, izolirati strukturne posebnosti ali napake pri pridobivanju podatkov. *Gephi* je orodje, ki dopolnjuje tradicionalno statistiko, saj vizualno razmišljanje z interaktivnimi vmesniki olajša argumentiranje in dokazovanje. Razvit je bil na raziskovalnem področju vizualne analitike.

Program *Gephi* omogoča več vrst postavitev vizualizacije grafov: OpenOrd, ForceAtlas, Yifan Hu, Frushterman-Reingold, Circular, Radial Axis in

GeoLayout. Postavitev izberemo glede na topologijo podatkov, ki jih hočemo predstaviti v obliki grafa. Kadar je poudarek na nastalih gručah, uporabimo OpenOrd. Algoritme ForceAtlas, Yifan Hu, Frushterman-Reingold uporabimo pri določanju lastnosti omrežja. Pri razvrščanju (angl. ranking) uporabimo postavitev Circular ali Radial Axis, za geografsko razdelitev pa GeoLayout. Program *Gephi* upravlja z vsemi vrstami omrežij (usmerjena, neusmerjena, mešana). Velikost omrežij je lahko do milijona vozlišč in povezav.

Knjižnica *Gephi* temelji na orodju NetBeans. Program in knjižnica imata vgrajen 3D mehanizem. *Gephi* bere več vrst podatkovnih datotek âĂŞ GEXF, GDF, GML, GraphML, Pajek NET, GraphViz DOT, CSV, UCINET DL, Tulip TPL, Netdraw VNA, Spreadsheet. Platforma *Gephi* je bila razvita na tehnološki univerzi v mestu CompiÃĺgne v Franciji. *Gephi* je bil izbran za Google Summer of Code v letih 2009, 2010, 2011, 2012 in 2013. Skupnost platforme *Gephi* jo predstavlja kot "Photoshop za grafe" (angl. like Photoshop for graphs) [27].

3.15 NetworkX

NetworkX je paket v programskem jeziku Python za ustvarjanje, upravljanje, spreminjanje in proučevanje sktruktur, dinamik in funkcij kompleksnih omrežij. S paketom *NetworkX* lahko naložimo in shranimo omrežja v večino podatkovnih oblik, ustvarimo številne vrste naključnih in navadnih omrežij, analiziramo strukturo omrežja, zgradimo modele omrežja, oblikujemo nove algoritme omrežja, narišemo omrežja in še veliko več.

Paket *NetworkX* zagotavlja:

- orodja za proučevanje strukture in dinamike omrežij;
- standardno programsko implementacijo vmesnika, ki je primerna za številne aplikacije;
- hiter razvoj okolja za skupinske multidisciplinarne projekte;
- vmesnik obstoječih numeričnih algoritmov in kode, napisani v programskih jezikih C, C++ ali FORTRAN;
- obravnave velikih množic podatkov.

Knjižnica omogoča pet različnih postavitev vizualizacije: krožna, naključna, v obliki lupine (angl. shell layout), z uporabo algoritma Fruchterman-Reingold, z uporabo lastnih vektorjev grafa Laplacian.

NetworkX je brezplačni paket. Zasnovan je bil leta 2002 in je trenutno pod licenco BSD. Prvotno različico so napisali Aric Hagberg, Dan Schult in Pieter Swart leta 2002 in 2003. Prva javna objava paketa je bila leta 2005. Paket ima bogato dokumentacijo z veliko primeri [28].

3.16 igraph

Igraph je brezplačni programski paket za ustvarjanje, upravljanje in spreminjanje usmerjenih in neusmerjenih grafov. Vključuje izvedbo klasičnih problemov teorije grafov, kot sta odkrivanje minimalnih vpetih dreves in pretoka v omrežju. Paket vsebuje algoritme sodobnih metod analize omrežij, kot je iskanje strukture skupnosti.

Učinkovita izvedba paketov *igraph* omogoča upravljanje grafov z milijoni vozlišči in povezavami. Skrito pravilo je, da če je graf ustrezen za fizični pomnilnik računalnika, je knjižnica *igraph* sposobna upravljati z njim.

Paket omogoča več vrst postavitve vizualizacije: krožna, rekurzivna, z uporabo algoritma Fruchterman-Reingold, z uporabo algoritma Kamada-Kawai, za velika omrežja, drevesna z uporabo algoritma Reingold-Tilford, sferična.

Knjižnico *igraph* je možno namestiti v več oblikah:

- *igraph* kot knjižnica C - za projekte napisane v programskem jeziku C ali C++ ali za izvedbo lastne analize omrežij ali modelov v jeziku C/C++ z uporabo podatkovnih struktur ali funkcij, ki jih vsebuje *igraph*.

- *igraph* kot paket R - uporabi se lahko kot dodaten paket projektu GNU R za statistično obdelavo podatkov. Fleksibilnost jezika R in njegovo bogastvo statističnih metod doda velik del produktivnosti knjižnice *igraph* z zelo majhno kaznijo pri hitrosti.

- *igraph* kot modul v programskem jeziku Python - pri tem načinu se lahko kombinira z velikim naborom funkcij in modulov, ki jih vsebuje jezik Python. Prednost je tudi enostavnost jezika Python.

- *igraph* kot dodatek za programski jezik Ruby.

Vsaka oblika paketa *igraph* v jedru vsebuje isto kodo napisano v jeziku ANSI C. Dokumentacija je na voljo za vsako različico knjižnice [29].

3.17 JUNG

JUNG (angl. Java Universal Network/Graph Framework) je programska knjižnica, ki zagotavlja enoten jezik za modeliranje, analizo in vizualizacijo podatkov, ki jih je mogoče predstaviti v obliki omrežja. Napisana je v programskem jeziku Java, kar omogoča *JUNG* aplikacijam uporabo bogatih vgrajenih možnosti Java API kot tudi drugih obstoječih knjižnic, napisanih v programskem jeziku Java.

Arhitektura knjižnice *JUNG* omogoča podporo različnih predstavitev entitet in relacij, usmerjenih in neusmerjenih grafov, grafov z vzporednimi povezavami in hipergrafov. Zagotavlja mehanizem za označevanje omrežij, entitet in relacij z metapodatki. Ta možnost olajša ustvarjanje analitičnih orodij za kompleksne množice podatkov, ki preučujejo relacije med entitetami ali metapodatki, ki so lahko priloženi k vsaki entiteti in relaciji.

Trenutna različica knjižnice *JUNG* vključuje implementacije številnih algoritmov teorije grafov, rudarjenja podatkov in analize družbenih omrežij, kot so na primer razvrščanje v gruče, optimizacije, ustvarjanje naključnih grafov, statistične analize in računanje razdalj med vozlišči.

Knjižnica *JUNG* vsebuje ogrodje za vizualizacijo, ki omogoča enostavno izdelavo orodij za interaktivno raziskovanje omrežja. Uporabniki lahko uporabijo enega izmed vgrajenih algoritmov za postavitev omrežja pri vizualizaciji (krožna, drevesna, z uporabo algoritmov Fruchterman-Reingold in Kamada-Kawai, radialno drevesna, v obliki balona) ali pa ustvarijo svojega. Poleg tega so na voljo mehanizmi, ki uporabnikom omogočajo usmerjanje pozornosti na določen del grafa.

JUNG je odprtokodna knjižnica, ki zagotavlja enotno ogrodje za analizo in vizualizacijo omrežij. Dokumentacija in priloženi primeri se nahajajo na spletni strani projekta [30].

3.18 Prefuse

Prefuse je nabor programskih orodij za ustvarjanje bogate interaktivne vizualizacije podatkov. Podpira velik nabor funkcij za modeliranje, vizualizacijo in interakcijo podatkov. Zagotavlja optimizirane podatkovne strukture za tabele, grafe in drevesa, vsebuje različne vrste postavitve in vizualnih tehnik in podpira animacijo, dinamične poizvedbe, integrirano iskanje in povezljivost baz podatkov. Vrste postavitev, ki so omogočeni v knjižnici *prefuse* so: krožna, drevesna, radialno drevesna, v obliki balona, z uporabo algoritma Fruchterman-Reingold.

Prefuse je zasnovan tako, da vizualizira medsebojno povezane informacije v obliki grafa ali drevesne strukture. Uporabijo se lahko tudi nepovezani podatki, vendar so shranjeni v podatkovni tabeli. Risanje vizualnih objektov se izvaja z uporabo prikazovalnika (angl. renderer), ki ima dostop do samega objekta in do konteksta pogleda Graphics2D. Ta pristop omogoča dostop do celotnega nabora razpoložljivih metod risanja programskega jezika Java, kar pomeni, da je risanje popolnoma neodvisno od samega orodja.

Poleg zagotavljanja velikega nabora vnaprej določenih elementov za vizualizacijo podatkov, je velik del pozornosti usmerjen k uporabnosti vizualizacije. Ta je dosežena z uporabo elementov za lažjo interakcijo: nasveti orodij (angl. tooltips) in vlečenje (angl. dragging) vizualnih elementov. V knjižnici so na voljo tudi delno ali v celoti bolj specifične tehnike, kot npr. možnosti povečevanja, horizontalno premikanje ali semantično povečevanje. *Prefuse* razlikuje absolutne koordinate in koordinate pogleda. Ta delitev pomaga uporabniku postaviti vizualne elemente na logični način brez upoštevanja kasneje uporabljenih vizualnih tehnik, ki spreminjajo celoten pogled.

Knjižnica *prefuse* je napisana v programskem jeziku Java, z uporabo grafične knjižnice Java2D, in lahko enostavno dopolni aplikacije Java Swing ali spletne javanske programe. Bere datoteke GraphML (XML), TreeML (XML), CSV in datoteke, kjer so podatki razmejeni s tabulatorjem. *Prefuse* je odprtokodna knjižnica dostopna tudi na spletni strani GitHub. Knjižnica je licencirana pod licenco BSD. Osnove knjižnice so bile postavljene v laboratoriju za vizualizacije na Univerzi Berkeley [31].

Poglavje 4

Podatki

Pri primerjavi orodij za vizualizacijo smo uporabili omrežja iz zbirke podatkov Pajek [32]. Uporabili smo omrežja treh različnih velikosti: majhno (okrog 10 vozlišč), srednje (okrog 7000 vozlišč) in veliko (okrog 80000 vozlišč). V primeru težav pri vizualizaciji največjega omrežja, smo pri analizi preizkusili še dodatno omrežje (okrog 20000 vozlišč).

Prvo omrežje se imenuje *Tina* in vsebuje 11 vozlišč in 41 usmerjenih povezav. Omrežje sestavlja komunikacijsko interakcijo med enajstimi člani in svetovalci v Študentskem svetu Univerze v Ljubljani. Rezultati meritev niso realne interakcije med akterji, ampak nabor znanja o komunikacijskih interakcijah. Podatki so zbrani leta 1992 na osebnih razgovorih s sodelujočimi člani. Člani skupine so bili vprašani glede opredelitve članov njihovih egocentričnih omrežij po spominu. Vsebina komunikacijskega toka je omejena na zadeve znotraj študentskega sveta. Časovni okvir raziskave je obdobje šestih mesecev (od oblikovanja sveta do razgovora) [33].

Drugo omrežje se imenuje *geom* in je sestavljeno iz 7343 vozlišč in 11898 neusmerjenih povezav. Podatki predstavljajo omrežje sodelovanja med avtorji na področju računske geometrije. Omrežje avtorjev je pridobljeno iz bibliografije BibTeX [34], iz podatkovne baze računske geometrije geombib [35]. Dva avtorja sta povezana, če sta soavtorja nekega dela (npr. članek, knjiga, revija). Povezave imajo še vrednost, ki predstavljajo število skupnih del.

Tretja zbirka podatkov (*hep-th* [36]) je usmerjeno omrežje z 27240 vozlišči in 342437 povezavami. Sestavljeno je iz citatov med članki iz spletne baze arXiv [37] na temo fizike visokoenergijckih delcev iz leta 2003. Imena vozlišč so identifikacijske številke člankov, povezava med vozliščema X in Y pa pomeni, da članek X citira Y.

Četrta in največja zbirka podatkov je velika baza angleškega jezika *Word-*

Net [38]. Omrežje je usmerjeno, sestavljeno iz 82670 vozlišč (besed) ter 133445 povezav, ki pomenijo sorodnost dveh besed. Samostalniki, glagoli, pridevniki in prislovi so združeni v množicah kognitivnih sinonimov (angl. synsets), kjer vsak izraža različen koncept. Sinseti so povezani med seboj glede na konceptualno-semantične in leksikalne relacije. Nastalo omrežje predstavlja smiselno povezane besede. *WordNet* je dejansko besednjak, ki združuje besede na podlagi njihovih pomenov. Uporabljena različica podatkovne zbirke je bila zajeta v formatu Pajek leta 2004.

Omrežja in njihove osnovne lastnosti so prikazane v tabeli 4.1. Uporabljen format podatkovnih datotek pri vseh omrežjih je Pajek (datoteke s končnico .net). Specifikacije strojnih komponent računalnika so sledeče: procesor Intel Core2Duo 2x2.20Ghz, pomnilnik 4GB, 64 bitni operacijski sistem Windows 7 Ultimate.

Tabela 4.1: Uporabljena omrežja.

Ime omrežja	Število vozlišč	Število povezav
Tina	11	41
geom	7343	11898
hep-th	27240	342437
WordNet	82670	133445

Poglavje 5

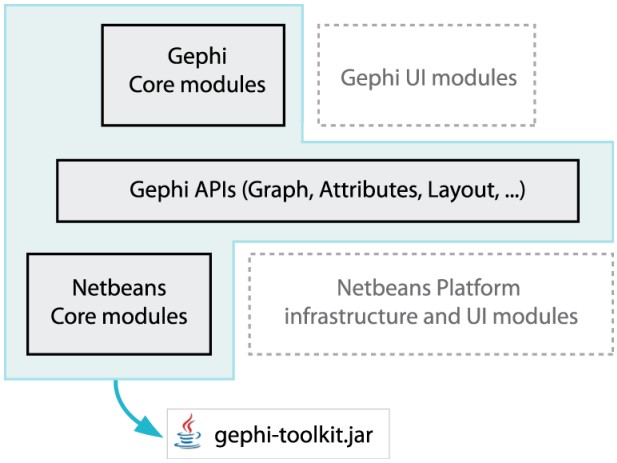
Analiza knjižnic in rezultati

V tem poglavju bomo opisali in analizirali pet knjižnic za vizualizacijo omrežij: *Gephi*, *NetworkX*, *igraph*, *JUNG* in *Prefuse*. Knjižnice so bile izbrane na podlagi dokumentacije, dostopnosti in programskega jezika. Pri vsaki knjižnici smo pogledali: (1) kako dodamo vozlišča in povezave v omrežje, (2) kako spreminjamo lastnosti vozlišč in povezav (barva, velikost, debelina), (3) kako dodamo vozliščem in povezavam atribute in kako se le-ti prikažejo, (4) kako velika omrežja lahko vizualiziramo.

5.1 Gephi

Paket programskih orodij *Gephi* (angl. toolkit) vsebuje module (Graph, Layout, Filters, IO itd.) v eni sami standardni knjižnici programskega jezika Java. Zbirka orodij je datoteka JAR, katero lahko uporabimo v projektu, ki temelji na programskem jeziku Java. Knjižnica *Gephi* je način uporabe funkcij programa *Gephi*, vendar s skriptnim jezikom. Paket združuje funkcije programa *Gephi* s prilagodljivostjo knjižnic. Paket *Gephi* je modularno zasnovan. Vse funkcije so zapakirane v različne module, na primer modul za strukturo grafov, modul za algoritme postavitve. Poleg tega so poslovni moduli ločeni od modulov uporabniških vmesnikov. Namen paketa *Gephi* je zapakirati jedrne module in odstraniti sloj uporabniškega vmesnika. Slika 5.1 prikazuje arhitekturo programa *Gephi*.

Paket *Gephi* temelji na osnovi platforme NetBeans, kar omogoča razvoj in izvajanje potekata v NetBeans IDE (angl. Integrated development environment), ki v celoti vključuje vsa orodja za razvijalce v platformi NetBeans. Platforma NetBeans je splošno ogrodje za aplikacije Swing (osnovni gradnik paketa Java GUI, angl. primary Java GUI widget toolkit) [39]. Različici



Slika 5.1: Arhitektura programa in programskega paketa *Gephi* [27].

obeh uporabljenih orodij sta *Gephi* 0.8.7 in NetBeans IDE 7.3.1 za 64 bitni operacijski sistem Windows.

5.1.1 Ustvarjanje omrežja

Preden začnemo uporabljati knjižnico je potrebno ustvariti projekt. Ustvarjanje novega projekta ustvari tudi delovno okolje (angl. workspace). Delovna okolja so vsebnik (angl. container) za vse podatke. Projekt lahko vsebuje več med seboj neodvisnih delovnih okolij. Moduli za delovanje pogosto potrebujejo delovno okolje, zato ga je potrebno ustvariti na samem začetku.

```
ProjectController pc = Lookup.getDefault().lookup(
    ProjectController.class);
pc.newProject();
Workspace workspace = pc.getCurrentWorkspace();
```

Koda 5.1: Inicializacija projekta in delovnega okolja.

Graph API je namenjen shranjevanju in upravljanju strukture grafov. Pri ročnem ustvarjanju vozlišč in povezav je potrebno vsakega izmed njih definirati. Pri tem lahko dodatno poimenujemo oznake vozlišč. Kadar ustvarimo povezavo, je potrebno navesti katera vozlišča povezuje. Poleg tega lahko povezavam dodamo utež in podatek o usmerjenosti povezave. Na koncu grafu dodamo vozlišča in povezave, pri čemer je potrebno navesti tudi tip grafa.

Primer 5.2 predstavlja ustvarjanje usmerjenjega grafa.

```
GraphModel graphModel = Lookup.getDefault().lookup(
    GraphController.class).getModel();

Node n0 = graphModel.factory().newNode("n0");
n0.getNodeData().setLabel("Node 0");
Node n1 = graphModel.factory().newNode("n1");
n1.getNodeData().setLabel("Node 1");

Edge e1 = graphModel.factory().newEdge(n1, n0, 1f, true);

DirectedGraph directedGraph = graphModel.getDirectedGraph();
directedGraph.addNode(n0);
directedGraph.addNode(n1);
directedGraph.addEdge(e1);
```

Koda 5.2: Ročno ustvarjanje grafa.

5.1.2 Spreminjanje lastnosti

Pri vizualizaciji omrežja je pomembno, na kakšen način se podatki izrišejo. Za boljše razumevanje stvari, ki jih želimo poudariti, je potrebno spremeniti lastnosti vozlišč in povezav glede na nek kriterij. Spremenimo lahko barvo in velikost vozlišč ter debelino povezav.

Knjižnica *Gephi* omogoča spreminjanje barve vozlišč in povezav. Pri spremembi barve in velikosti vozlišč lahko uporabimo vgrajen API za razvrščanje. Pri tem izkoristimo na primer različne lastnosti vozlišč ali omrežja. V našem primeru barvo določimo glede na stopnjo vozlišča (koliko povezav gre v in iz vozlišča), velikost pa glede na središčnost. Pri izbiri velikosti nastavimo razpon (največjo in najmanjšo velikost), v katerem se vozlišča izrišejo.

```
Ranking degreeRanking = rankingController.getModel().getRanking
    (Ranking.NODE_ELEMENT, Ranking.DEGREE_RANKING);
AbstractColorTransformer colorTransformer = (
    AbstractColorTransformer) rankingController.getModel().
    getTransformer(Ranking.NODE_ELEMENT, Transformer.
    RENDERABLE_COLOR);
colorTransformer.setColors(new Color[]{new Color(0xFEF0D9), new
     Color(0xB30000)});
rankingController.transform(degreeRanking,colorTransformer);

AttributeColumn centralityColumn = attributeModel.getNodeTable
```

```
    ().getColumn(GraphDistance.BETWEENNESS);
Ranking centralityRanking = rankingController.getModel().
    getRanking(Ranking.NODE_ELEMENT, centralityColumn.getId());
AbstractSizeTransformer sizeTransformer = (
    AbstractSizeTransformer) rankingController.getModel().
    getTransformer(Ranking.NODE_ELEMENT, Transformer.
    RENDERABLE_SIZE);
sizeTransformer.setMinSize(3);
sizeTransformer.setMaxSize(10);
rankingController.transform(centralityRanking,sizeTransformer);
```

Koda 5.3: Spreminjanje barve in velikost vozlišč.

Barvo in debelino povezav nastavimo enostavno z dvema ukazoma v modelu vgrajenega API PreviewModel, ki je bil definiran na samem začetku programa.

```
model.getProperties().putValue(PreviewProperty.EDGE_COLOR, new
    EdgeColor(Color.RED));
model.getProperties().putValue(PreviewProperty.EDGE_THICKNESS,
    new Float(0.1f));
```

Koda 5.4: Nastavitev barve in debeline povezav.

5.1.3 Dodajanje atributov

Atribute lahko dodamo vozliščem in povezavam. Oznake vozlišč lahko poimenujemo že pri ročnem ustvarjanju vozlišč, kot smo pokazali v kodi 5.2. Naknadno pa lahko dodamo stolpce z dodatnimi atributi. Povezavam ročno dopišemo, kakšno utež imajo (primer v kodi 5.2) ali uporabimo takšno, ki je navedena v podatkih. Prikaz in uporabo atributov lahko na enostaven način vključimo v vizualizacijo. Knjižnica *Gephi* ima prav tako omogočeno nastavitev za prikaz oznak sorazmerno z velikostjo vozlišč, če velikost vozlišč ni določena po drugih kriterijih.

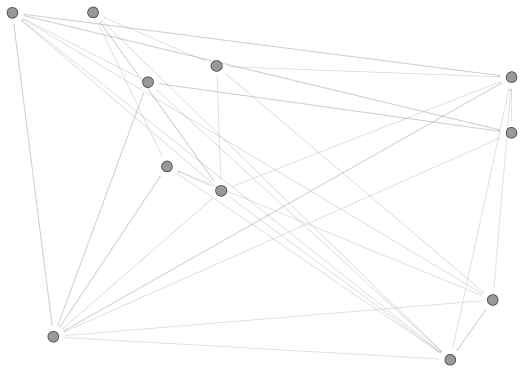
```
PreviewModel previewModel = Lookup.getDefault().lookup(
    PreviewController.class).getModel();
previewModel.getProperties().putValue(PreviewProperty.
    SHOW_NODE_LABELS, Boolean.TRUE);
previewModel.getProperties().putValue(PreviewProperty.
    NODE_LABEL_PROPORTIONAL_SIZE, Boolean.TRUE);
```

Koda 5.5: Prikaz in nastavitev oznak.

5.1.4 Vizualizacija

V nadaljevanju smo preizkusili vizualizacijo izbranih omrežij. Pri tej knjižnici je vizualizacija omogočena s privzetim razredom JFrame programskega jezika Java. Razred, ki poskrbi za risanje samega omrežja, se imenuje *PApplet* in je del programskega jezika Processing.

Pri omrežju *Tina* (slika 5.2), se vidijo vsa vozlišča in vse povezave. Slabo se vidi usmerjenost omrežja, ker so puščice na koncu povezav zelo majhne. Privzeta barva risanja vozlišč in povezav je siva. Nastaviti je bilo potrebno barvo ozadja, v tem primeru je bela. Privzeti način risanja povezav je s krivuljami, namesto z ravnimi črtami. V našem primeru je ta možnost onemogočena.



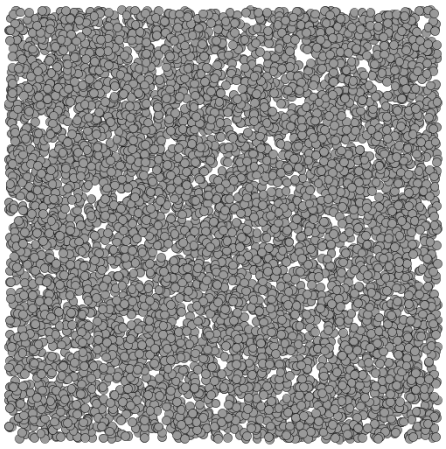
Slika 5.2: Vizualizacija omrežja *Tina* s pomočjo knjižnice *Gephi*.

Pri vizualizaciji srednjega (slika 5.3) in velikega omrežja (slika 5.4) je težko opaziti posamezne povezave med vozlišči. Še posebej zaradi (privzete) sive barve vozlišč. Obe omrežji imata preveč vozlišč za učinkovit prikaz. Brez dodatnega povečevanja ali uporabe dodatnih algoritmov, je s slike težko karkoli razbrati.

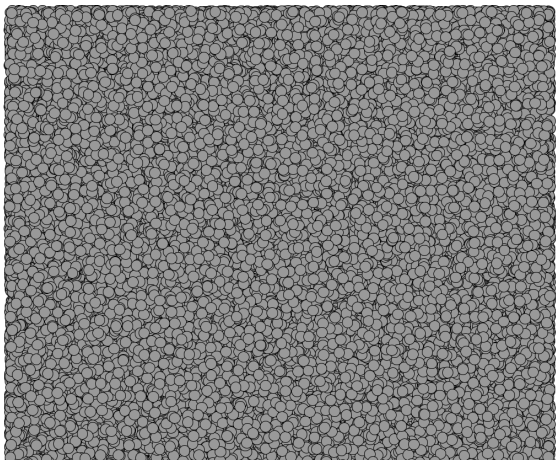
Pri vizualizaciji omrežja *WordNet*je bilo najprej potrebno nastaviti višjo razpoložljivost računalniškega spomina v programu NetBeans. Pri vsakem prikazu omrežja na zaslonu je bilo potrebno klikniti v okno JFrame, sicer je bilo okno prazno.

5.2 NetworkX

NetworkX je paket programskega jezika Python za ustvarjanje, upravljanje, spreminjanje in analizo kompleksnih omrežij. Strukturo knjižnice *NetworkX* je



Slika 5.3: Vizualizacija omrežja *geom* s pomočjo knjižnice *Gephi*.



Slika 5.4: Vizualizacija omrežja *WordNet* s pomočjo knjižnice *Gephi*.

mogoče videti v organizaciji izvorne kode. Paket zagotavlja razrede za objekte grafov, generatorje za ustvarjanje standardnih grafov, V/I (angl. IO, input/output) rutine za branje obstoječih podatkovnih zbirk, algoritme za analizo nastalih omrežij in nekaj osnovnih orodij za risanje grafov.

Večina vmesnikov paketa *NetworkX* je opremljena s funkcijami, ki vzamejo objekt grafa kot argument. Metode objektov grafa so omejene na osnovne manipulacije in prikaze. Ta lastnost zagotavlja modularnost kode in dokumentacije. Prav tako omogoča neizkušenim uporabnikom lažje razumevanje paketa.

Izvorna koda vsakega modula je napisana tako, da omogoča enostavno in razumljivo branje, s čemer lahko izvemo več o samih algoritmih, ki se izvajajo pri analizi omrežja.

Pri testiranju smo namestili ogrodje Python (x, y), različica 2.7.5.1, ki nam je omogočilo uporabo primernega razvijalnega okolja Spyder. Različica uporabljenega jezika Python je 2.7.5 na 32 bitnem operacijskem sistemu Windows, knjižnica *NetworkX* pa je 1.8.1.

5.2.1 Ustvarjanje omrežja

Preden začnemo uporabljati knjižnico *NetworkX*, je potrebno uvoziti modul v projek. Knjižnica ima implementirane štiri razrede za različne tipe grafov: usmerjen, neusmerjen, multigraf (graf, ki omogoča več povezav med dvema vozliščema) z usmerjenimi in neusmerjenimi povezavami. Vozlišča in povezave začnemo dodajati, ko imamo ustvarjen prazen graf. Pri knjižnici *NetworkX* so vozlišča lahko katerikoli razpršen (angl. hashable) objekt, npr. niz besedila, slika, XML objekt, drug graf itd. Vozlišča lahko dodajamo vsakega posebej ali jih definiramo pri ustvarjanju povezav. Povezavam lahko kot atribut dodamo utež.

```
import networkx as nx
G=nx.Graph()
G.add_node(1)
G.add_edge(1,2)
G.add_edge(2,3,weight=0.9)
```

Koda 5.6: Ročno ustvarjanje grafa.

Vozlišča lahko v graf dodamo tudi s seznamom, slovarjem, množico, drugim grafom, posebno datoteko itd. Povezave lahko dodamo eno za drugo ali jih definiramo kot seznam ali slovar.

5.2.2 Spreminjanje lastnosti

Knjižnica *NetworkX* ni zasnovana kot orodje za prikaz omrežij, zato je risanje zagotovljeno s preprostim vmesnikom za risanje, ki omogoča tudi več vrst postavitev. Na voljo je več vrst postavitve orodja *Graphiviz*, kot sta na primer *neato* in *dot*, s priporočenim paketom risanja *pygraphviz* ali *pydot*. Risanje je možno tudi s pomočjo zunanjih programov ali s paketom *Matplotlib* programskega jezika Python. Lastnosti vozlišč in povezav spreminjamo na enostaven način že pri samem izrisu omrežja. Z enim ukazom risanja lahko določimo

barvo, velikost in obliko vozlišč, barvo, debelino in stil povezav (polne, prekinjene, točkaste črte).

```
nx.draw(G,pos,node_color='blue',node_size=800,node_shape='s',
    edge_color='red',width=2.5,style='dashed')
```

Koda 5.7: Spreminjanje lastnosti vozlišč in povezav grafa.

5.2.3 Dodajanje atributov

Atribute kot so uteži, oznake, barve ali katerikoli drugi objekt, je mogoče dodati grafu, vozliščem ali povezavam. Vsak graf, vozlišče in povezava lahko vsebuje pare atributov ključ/vrednost, ki so shranjeni v slovarju atributov. V privzetem načinu so le-ti prazni, vendar je možno njihovo naknadno dodajanje ali spreminjanje z uporabo ukazov *add_ edge* in *add_ node* ali direktna manipulacija slovarjev atributov imenovanih *G.graph*, *G.node* in *G.edge* za graf G, pri čemer ti ukazi še ne dodajo komponente na graf.

```
G = nx.Graph(day="Friday")

G.add_node(1, name='Alice')
G.add_node(2, name='Bob')
G.node[1]['age'] = 25
G.node[2]['age'] = 31

G.add_edge(1,2,color='red',weight=0.84,size=300)
```

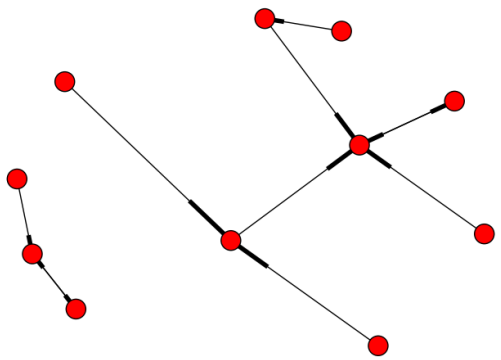
Koda 5.8: Dodajanje atributov grafu, vozliščem in povezavam.

Podatkovne strukture z opisom vozlišč lahko uporabimo le pod pogojem, da so razpršeni objekti. Če niso, lahko ustvarimo enolično identifikacijsko oznako, ki določa vozlišča in šele potem dodelimo podatke kot atribute. Če so podatki, ki opisujejo povezave, v številčni obliki in imajo namen predstaviti utežen graf (angl. weighted graph), uporabimo besedo *weight* kot ključ atributa. Atribute vozlišč in povezav lahko uporabimo in prikažemo pri risanju grafa. Pri tem lahko določimo tudi velikosti, barve in debeline pisave črk v oznakah.

5.2.4 Vizualizacija

Izbrana omrežja smo vizualizirali s pomočjo dodatne knjižnice *Matplotlib* programskega jezika Python. V privzetem načinu risanja so oznake vozlišč prika-

zane, v našem primeru je ta možnost onemogočena. Privzeti barvi vozlišč in povezav sta rdeča in črna. Pri omrežju *Tina* (slika 5.5), se vidijo vsa vozlišča in tudi povezave med njimi. Puščiče, ki kažejo usmerjenost grafa, se jasno vidijo pri vsaki povezavi.

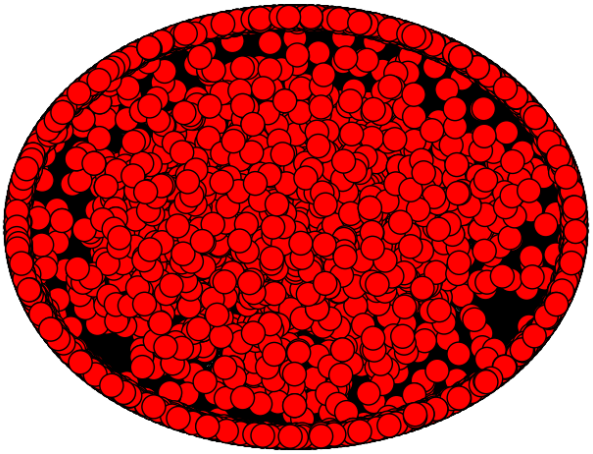


Slika 5.5: Vizualizacija omrežja *Tina* s pomočjo knjižnice *NetworkX*.

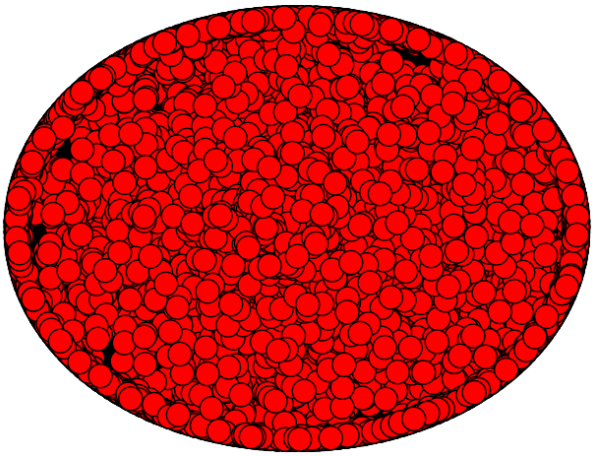
Opazimo, da je narisano omrežje drugačno kot pri knjižnici *Gephi*. Ugotovili smo, da knjižnica *NetworkX* bere format datotek Pajek z ukazom *read_pajek* vedno kot seznam povezav in pričakuje, da so povezave napisane v parih v vsaki vrstici. Uradni format datotek Pajek omogoča različne možnosti zapisa omrežja: seznam povezav, seznam sosednosti in matrika sosednosti. V našem primeru imamo omrežje *Tina* zapisano v obliki seznama sosednosti. Za uspešno branje takega formata je za knjižnico *NetworkX* potrebno spremeniti .net podatke v .ajlist, izbrisati oznake vozlišč in opustiti seznam povezav s številkami vozlišč ali pa ustvariti seznam povezav z oznakami vozlišč. Zadnja možnost nam omogoča ohranitev oznake vozlišč. Spremenjene podatke je v tem primeru potrebno naložiti z ukazom *read_adjlist*.

Vizualizacija srednjega omrežja je prikazana na sliki 5.6. V tem primeru je nemogoče določiti, katera vozlišča so povezana med seboj. Privzeta oblika risanja je s postavitvijo usmerjenih sil (*spring_layout*).

Risanje omrežja *WordNet* je bilo neuspešno, zaradi šibkih specifikacij računalnika, zato smo vizualizirali tretjo zbirko podatkov, ki ima nekaj tisoč vozlišč in povezav manj. Na sliki 5.7 se opazi, da je težko prepoznati povezave med vozlišči.



Slika 5.6: Vizualizacija omrežja *geom* s pomočjo knjižnice *NetworkX*.



Slika 5.7: Vizualizacija omrežja *hep-th* s pomočjo knjižnice *NetworkX*.

5.3 igraph

Knjižnica *igraph* je brezplačen paket programskega jezika Python. Programski paket vsebuje priročna orodja za analizo omrežij. Knjižnica *igraph* je odprtokodna in sposobna upravljati z velikimi grafi, ki imajo lahko milijone vozlišč in povezav. Vsebuje funkcije za ustvarjanje, upravljanje, spreminjanje in vizualizacijo omrežij, računanje različnih strukturnih lastnosti, omogoča uvoz in zapis več različnih formatov datotek. S pomočjo vmesnika visokonivojskega

programskega jezika Python podpira hiter razvoj in ustvarjanje prototipov.

Knjižnica *igraph* vsebuje funkcije za ustvarjanje navadnih in naključnih grafov z uporabo več algoritmov in modelov iz teorije omrežij. Zagotavlja enostavnost pri upravljanje grafov, dodajanje in brisanje vozlišč in povezav. Knjižnica je opremljena s podatkovnimi tipi, s katerimi je možno implementirati lastne algoritme v programskih jezikih C, R, Python in Ruby.

Uporabljena različica knjižnice *igraph* je 0.6.5 za že obstoječo različico programskega jezika Python 2.7 za 32 bitni operacijski sistem Windows. Za uvoz paketa *igraph* smo uporabili orodje PyCharm IDE 3.0.2.

5.3.1 Ustvarjanje omrežja

Preden začnemo uporabljati knjižnico *igraph*, jo je potrebno pravilno uvoziti. Naslednji korak je ustvarjanje grafa, ki je na začetku prazen. V graf lahko dodamo vsako vozlišče posebej ali po več naenkrat. Za povezave velja isto pravilo, le da je potrebno določiti, kateri dve vozlišči povezuje, kar naredimo s seznamom indeksev vozlišč.

```
from igraph import *
g = Graph()
g.add_vertices(3)
g.add_edges([(0,1), (1,2), (2,0)])
```

Koda 5.9: Ročno ustvarjanje grafa, dodajanje vozlišč in povezav.

5.3.2 Spreminjanje lastnosti

Prikazovanje in risanje grafov sta pri paketu *igraph* odvisna od knjižnice *pycairo* programskega jezika Python. Knjižnica *igraph* ima bogato izbiro možnosti spreminjanja oblike grafa. Obstajajo spremenljivke za spreminjanje velikosti, barve, oblike, debeline in barve okvirja vozlišč kot tudi barve in debeline povezav. Na voljo so tudi možnosti za upravljanje z oznakami vozlišč in s puščicami povezav. Z enim ukazom lahko določimo vse te parametre ali ustvarimo slovar le-teh.

```
plot(g, vertex_size=20, vertex_color="blue", edge_width=2.5,
    edge_color="red", edge_curved=True)
```

Koda 5.10: Spreminjanje lastnosti vozlišč in povezav.

5.3.3 Dodajanje atributov

Knjižnica *igraph* uporablja identifikacijske številke pri označevanju vozlišč in povezav. Oštevilčenje se začne z nič in se ohrani ves čas obstoja grafa (če vozlišče ali povezavo odstranimo, se oznake avtomatsko zamaknejo, da so ves čas med nič in številom vozlišč v omrežju). V primeru, da želimo vozliščem dodati atribute oziroma imena, jih definiramo kot seznam. Seznam je potrebno obnavljati po vsaki spremembi v grafu. Vozlišča in povezave so zgrajene kot slovar, kar nam omogoča dodajanje atributov v stilu ključ/vrednost, pri katerem je ključ ime atributa (edini pogoj je, da mora biti tipa niz), vrednost pa je sam atribut. Ročno ustvarimo graf in dodamo ime in starost vsakemu vozlišču kot atribut, povezavam pa shranimo uteži. Atribute lahko potem uporabimo pri risanju grafa in določanju barv vozlišč in povezav. Kadar uporabimo *vs* in *es* slovarja, dodamo atribute vsem vozliščem in povezavam v grafu.

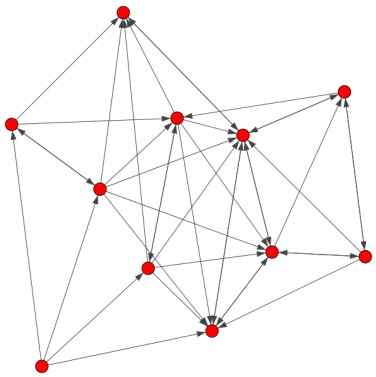
```
g = Graph([(0,1), (0,2))
g.vs["name"] = ["Alice", "Bob"]
g.vs["age"] = [25, 31]
g.es["weight"] = [0.84, 0.5]
```

Koda 5.11: Dodajanje atributov vozliščem in povezavam.

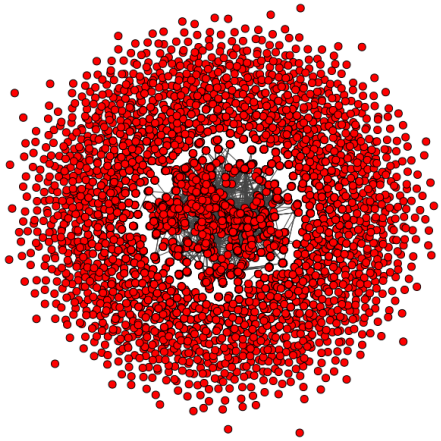
5.3.4 Vizualizacija

Pri vizualizaciji naših podatkov smo uporabili dodatno knjižnico *pycairo* programskega jezika Python. V privzetem načinu so vozlišča v rdeči barvi, povezave pa v sivi. Pri vizualizaciji omrežja *Tina* (slika 5.8) smo v prvem poskusu dobili narisano neusmerjeno omrežje. Nato smo izbrisali komentar, ki se je nanašal na morebitne neusmerjene povezave. V našem primeru je omrežje usmerjeno, kot je bilo razvidno tudi iz podatkov. Kadar ni eksplicitno navedeno, so povezave v obliki krivulje. V našem primeru smo to onemogočili, zato so vozlišča povezana z ravnimi črtami.

Pri vizualizaciji srednjega (slika 5.9) in največjega omrežja (slika 5.10) je nastal problem pri branju podatkov. V omrežju *geom* so bile podane koordinate vozlišč (x, y, z), vendar so bile vse vrednosti pri vsakem vozlišču enake, zato jih je bilo potrebno odstraniti. V četrti zbirki smo zamenjali podatek o usmerjenosti povezav, da smo dobili neusmerjeno omrežje. Pri obeh vizualizacijah je brez dodatnega popravljanja podatkov knjižnica *igraph* izrisala eno vozlišče.



Slika 5.8: Vizualizacija omrežja *Tina* s pomočjo knjižnice *igraph*.

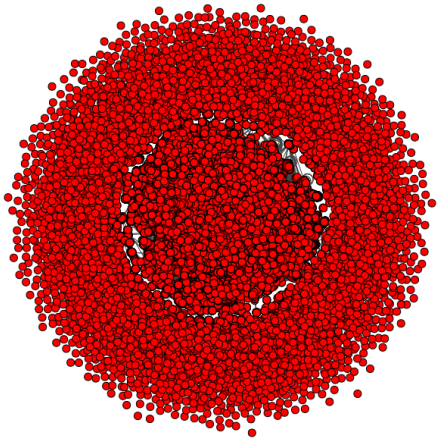


Slika 5.9: Vizualizacija omrežja *geom* s pomočjo knjižnice *igraph*.

5.4 JUNG

JUNG je odprtokodna knjižnica za modeliranje, analizo in vizualizacijo grafov in omrežij. Knjižnica je zasnovana v programskem jeziku Java, kar omogoča uporabo vgrajenih funkcij iz Java API in tudi drugih zunanjih knjižnic tega jezika. *JUNG* podpira večino vrst omrežij. Knjižnica vsebuje bogato strukturo atributov in ravna z omrežji na enostaven način. Ima vgrajene predikate, kot so množice, podmnožice, filtre, postavitve itd. Knjižnica *JUNG* vsebuje obsežne in razvijajoče se knjižnice algoritmov.

JUNG je ogrodje, na katerem zgradimo aplikacije in orodja za upravljanje



Slika 5.10: Vizualizacija omrežja *WordNet* s pomočjo knjižnice *igraph*.

in spreminjanje grafov. Uporablja se lahko pri enostavnih primerih kode za testiranje ali kot podpora v razvoju zapletenih orodij z grafičnim uporabniškim vmesnikom.

Za uporabo ogrodja *JUNG* je bilo potrebno namestiti Maven 3.0.4 (orodje, ki se uporablja za izgradnjo in upravljanje vsakega projekta, ki temelji na programskem jeziku Java) v orodju Eclipse IDE. Pri našem testiranju je bila usposobljena različica knjižnice *JUNG2* 2.0.1, različica programskega jezika Java pa je Java 1.7.

5.4.1 Ustvarjanje omrežja

Ročno ustvarjanje grafa začnemo z definicijo tipa grafa, vozlišč in povezav. V našem primeru je graf tipa *SparseMultiGraph*, kar pomeni, da je lahko usmerjen, neusmerjen ali z vzporednimi povezavami. Vozlišča so tipa Integer, povezave pa tipa String. Pri dodajanju vozlišč določimo samo tip, pri povezavah pa povemo ime povezave, kateri dve vozlišči povezuje in tip - usmerjena ali neusmerjena (privzeto je neusmerjena).

```
Graph<Integer, String> g = new SparseMultigraph<Integer, String
    >();
g.addVertex((Integer)1);
g.addVertex((Integer)2);
g.addVertex((Integer)3);
g.addEdge("Edge-A", 1, 2);
```

```
g.addEdge("Edge-B", 2,3, EdgeType.DIRECTED);
```

Koda 5.12: Ročno ustvarjanje grafa.

5.4.2 Spreminjanje lastnosti

Pri spreminjanju barve vozlišč in povezav uporabimo razred *Transformer*, ki je del ogrodja Apache Commons Collections. Bistvenega pomena pri določanju barve in tipa povezav (polne, prekinjene) je orodje AWT (angl. Abstract Window Toolkit) programskega jezika Java.

```
Transformer<Integer,Paint> vertexPaint = new Transformer<
    Integer,Paint>() {
        public Paint transform(Integer i) {
                return Color.BLUE;
        }
};
Transformer<String, Paint> edgePaint = new Transformer<String,
    Paint>() {
        public Paint transform(String s){
                return Color.RED;
        }
};
```

Koda 5.13: Spreminjanje barve vozlišč in povezav.

Če izberemo tip prekinjene povezave, lahko določimo tudi dolžino neprekinjenega dela. Potem s pomočjo razreda *Transformer* uveljavimo spremembo. Na koncu moramo dodati vse nastavitve v panel prikaza *RenderContex*.

```
float dash[] = {20.0f};
final Stroke edgeStroke = new BasicStroke(2.0f, BasicStroke.
    CAP_BUTT, BasicStroke.JOIN_MITER, 10.0f, dash, 0.0f);
Transformer<String, Stroke> edgeStrokeTransformer = new
    Transformer<String, Stroke>() {
        public Stroke transform(String s) {
                return edgeStroke;
        }
};
vv.getRenderContext().setEdgeStrokeTransformer(
    edgeStrokeTransformer);
```

Koda 5.14: Spreminjanje tipa povezav in dodajanje na panel.

5.4.3 Dodajanje atributov

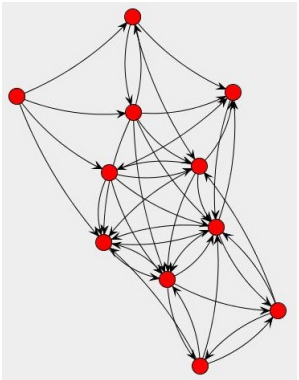
Povezave lahko poimenujemo že ob samem ustvarjanju, pod pogojem, da so tipa String. Enako velja za vozlišča, če pripadajo istemu tipu. Oznakam vozlišč lahko določimo poljuben položaj, oznake pa lahko pri vizualizaciji dodamo tudi povezavam.

```
g.addVertex("Square");
g.addVertex("Rectangle");
g.addEdge("Edge1", "Square", "Rectangle");
...
vv.getRenderer().getVertexLabelRenderer().setPosition(Position.
    CNTR);
vv.getRenderContext().setEdgeLabelTransformer(new
    ToStringLabeller());
```

Koda 5.15: Oznake vozlišč in povezav.

5.4.4 Vizualizacija

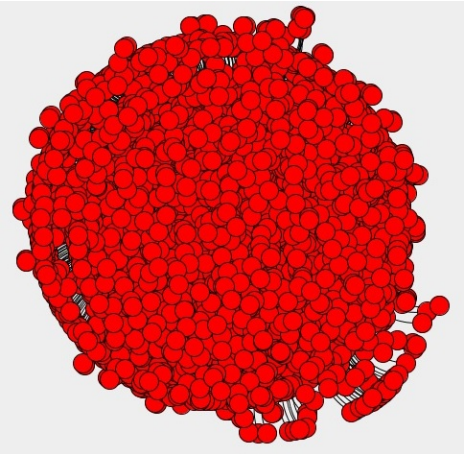
Za vizualizacijo smo spet uporabili JFrame programskega jezika Java. Knjižnica je imela probleme pri branju podatkov: če je bil kakšen podatek zapisan kot komentar, je vrnila napako. Potrebni so bili manjši popravki v podatkih. Pri risanju prve zbirke (slika 5.11) je lepo vidno, da je omrežje usmerjeno in katera vozlišča so med seboj povezana. Privzeti barvi risanja sta rdeča in črna.



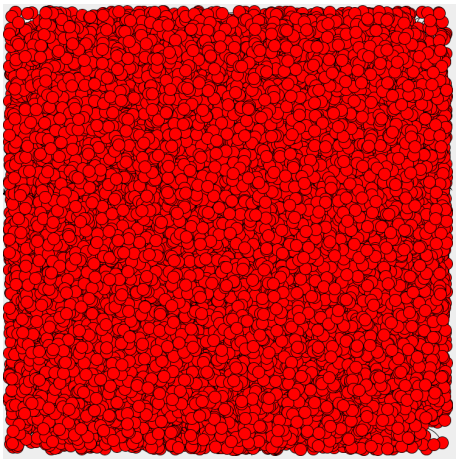
Slika 5.11: Vizualizacija omrežja *Tina* s pomočjo knjižnice *JUNG*.

Pri vizualizaciji druge (slika 5.12) in četrte največje (slika 5.13) zbirke po-

datkov je težko napovedati oziroma razbrati kaj s slike. Število vozlišč namreč vpliva na enostaven pogled na omrežje in brez dodatnega povečevanja slike ali uporabe dodatnih algoritmov, je to nemogoče. Knjižnica je uspešno prebrala in vizualizirala omrežje z več kot 80000 vozlišči.



Slika 5.12: Vizualizacija omrežja *geom* s pomočjo knjižnice *JUNG*.

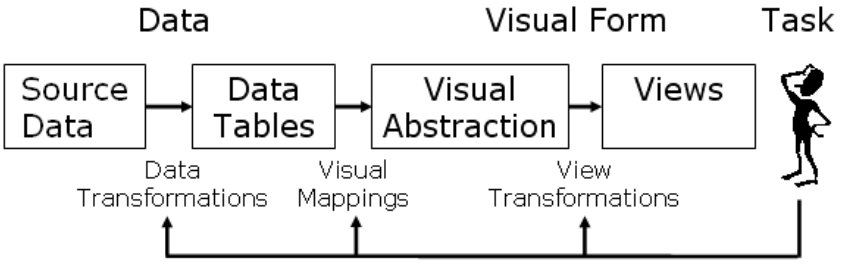


Slika 5.13: Vizualizacija omrežja *WordNet* s pomočjo knjižnice *JUNG*.

5.5 Prefuse

Prefuse je knjižnica, napisana v programskem jeziku Java z uporabo grafične knjižnice Java2D. *Prefuse* je paket orodij za vizualizacijo interaktivnih informacij. Knjižnica vsebuje podatkovne strukture in algoritme, sestavljene module, arhitekturne tehnike za skalabilnost in ima omogočeno animacijo.

Arhitektura knjižnice *Prefuse* upošteva cevovodno vizualizacijo (angl. visualization pipeline) (slika 5.14). Vir podatkov (angl. source data) so podatki, ki so osnova aplikacije. Ponavadi je vir podatkov ena datoteka, je pa možna uporaba tudi podatkovnih baz ali spletnih vsebin. Tabela podatkov (angl. data tables) se uporablja za shranjevanje abstraktnih podatkov. Vsaka vrstica tabele vsebuje podatkovni zapis in vsak stolpec vsebuje vrednosti za imenovano podatkovno polje z določenim tipom podatkov. Vsak zapis v tabeli je naveden kot n-terka (angl. tuple). Vizualna abstrakcija (angl. visual abstraction) vsebuje vizualne strukture, ki nastanejo s filtriranjem abstraktnih podatkov. V primeru vizualizacije, ki prikazuje vse elemente od začetka dalje, se filtriranje izvrši samo enkrat v času inicializacije. Pogledi (angl. views) shranjujejo slike vizualizacije.



Slika 5.14: Cevovodna arhitektura knjižnice *Prefuse* [31].

Uporabljena različica knjižnice *Prefuse* je prenešena s spletne strani GitHub in je uvožena v orodje Eclipse IDE.

5.5.1 Ustvarjanje omrežja

Preden ustvarimo graf je potrebno definirati tabelo, ki vsebuje vozlišča in povezave. Sledi definicija stolpcev s podatki. Pri povezavi ustvarimo dva stolpca, v katerih povemo, kateri dve vozlišči povezujemo. Nato dodamo vse v graf.

Če potrebujemo več podatkov, dodamo več stolpcev v tabelo. Primer 5.16 prikazuje ročno ustvarjanje usmerjenega grafa.

```
Table nodeData = new Table();
Table edgeData = new Table();
nodeData.addColumn("name", String.class);
edgeData.addColumn(Graph.DEFAULT_SOURCE_KEY, int.class);
edgeData.addColumn(Graph.DEFAULT_TARGET_KEY, int.class);

Graph graph = new Graph(nodeData, edgeData, true);
Node n1 = graph.addNode();
Node n2 = graph.addNode();
Node n3 = graph.addNode();
Edge e1 = graph.addEdge(n1, n2);
Edge e2 = graph.addEdge(n1, n3);
```

Koda 5.16: Ročno ustvarjanje usmerjenega grafa.

5.5.2 Spreminjanje lastnosti

Barve vozlišč in povezav se enostavno spremeni z uporabo vgrajenega razreda *ColorAction* knjižnice *Prefuse*. Po tej definiciji je potrebno ustvariti seznam procesov (angl. action list), ki vsebuje vse dodelitve barv. Seznam procesov se uporablja za učinke, ki se zgodijo v istem trenutku. Možna je tudi sprememba barve puščic usmerjenih povezav.

```
ColorAction fill = new ColorAction("graph.nodes", VisualItem.
    FILLCOLOR, ColorLib.rgb(0, 0, 200));
ColorAction edges = new ColorAction("graph.edges", VisualItem.
    STROKECOLOR, ColorLib.rgb(200, 0, 0));
ColorAction arrow = new ColorAction("graph.edges", VisualItem.
    FILLCOLOR, ColorLib.gray(200));

ActionList color = new ActionList();
color.add(fill);
color.add(text);
color.add(edges);
color.add(arrow);
```

Koda 5.17: Spreminjanje barve vozlišč in povezav.

5.5.3 Dodajanje atributov

Atribute dodamo vozliščem in povezavam pri sami definiciji ali pa jih naknadno dodamo v tabelo. Dodamo lahko tudi poljubno število dodatnih atributov. Oznake vozlišč se prilagodijo glede na samo velikost vozlišč, kar pomeni, da je vozlišče veliko toliko, kolikor je velika njegova oznaka. To možnost lahko onemogočimo ali jo dodamo kot oznako indeksa vozlišča. Oznake povezav ni možno prikazati.

```
nodeData.addColumn("name", String.class);
n1.setString("name", "Node 1");
n2.setString("name", "Node 2");
n3.setString("name", "n3");
```

Koda 5.18: Ročno dodajanje imen vozlišč.

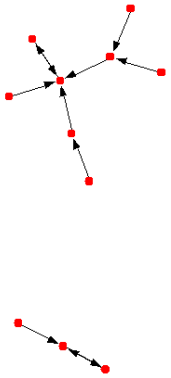
5.5.4 Vizualizacija

Vizualizacija omrežij s knjižnico *Prefuse* je omogočena na zelo specifičen način. Prvi korak je branje podatkov. *Prefuse* nima možnosti branja formata datotek Pajek. Preko knjižnice *NetworkX* smo z ukazom *write_graphml* spremenili podatke tipa .net v tip .gml. Drugi korak je vizualizacija. Dodamo graf in vse njegove lastnosti razredu *Visualization*. Pri tretjem koraku definiramo obliko vozlišč in privzeti prikazovlanik (angl. Default Renderer). Četrti korak vsebuje vse sezname procesov - določanje barv, način postavitve. V petem koraku določimo vizualni prikaz in interaktivne kontrole. Tukaj nastavimo velikost okna, dodamo možnost povečevanja in vlečenje grafa. V zadnjem koraku dodamo vizualni prikaz v JFrame. Vsi koraki so obvezni, kadar želimo vizualizirati omrežje.

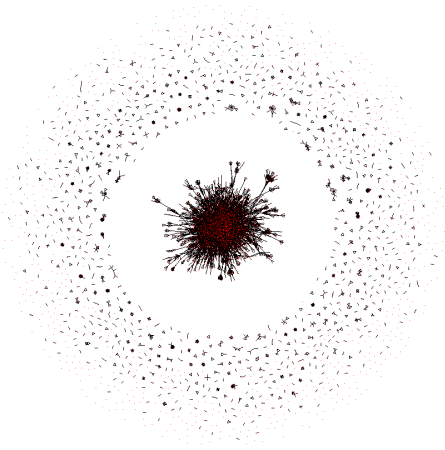
Pri vizualizaciji prvega omrežja (slika 5.15) je lepo razvidno, katera vozlišča so med seboj povezana s kakšno povezavo. Kot je že zgoraj navedeno, je bilo potrebno nastaviti barvo vozlišč, povezav in puščic, sicer je prikaz nemogoč.

Pri vizualizaciji omrežja *Tina* (slika 5.15) smo dobili isto omrežje, kot pri knjižnici *NetworkX*. Enakost je logična, saj smo pri pretvorbi podatkov iz .net v .gml uporabili najprej ukaz *read_pajek* in potem *write_graphml*.

V knjižnici *Prefuse* je pri vizualizaciji grafov potrebno vedno navesti vrsto postavitve. V našem primeru smo pri vsaki zbirki podatkov uporabili postavitev usmerjenih sil in sicer *ForceDirectedLayout*. Seveda se najbolj jasno ta postavitev vidi pri vizualizaciji srednjega (slika 5.16) in velikega omrežja.



Slika 5.15: Vizualizacija omrežja *Tina* s pomočjo knjižnice *Prefuse*.

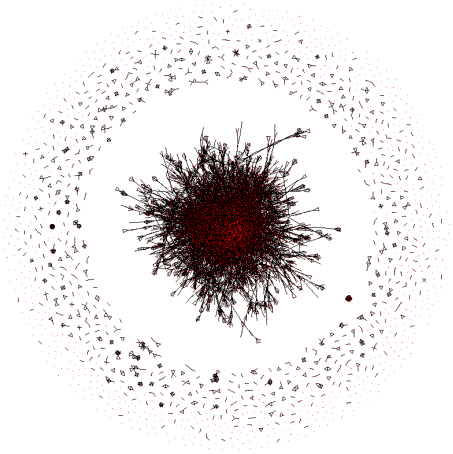


Slika 5.16: Vizualizacija omrežja *geom* s pomočjo knjižnice *Prefuse*.

Pri branju podatkov največje zbirke podatkov se je pojavil problem pomanjkanja spomina, zato smo izrisali omrežje *hep-th* (slika 5.17).

5.6 Diskusija

Pri primerjavi petih knjižnic smo se osredotočili na štiri točke branja podatkov ter vizualizacije omrežij: (1) kako dodamo vozlišča in povezave v omrežje, (2) kako spreminjamo lastnosti vozlišč in povezav (barva, velikost, debelina), (3) kako dodamo vozliščem in povezavam atribute in kako se nam le-ti prikažejo,



Slika 5.17: Vizualizacija omrežja *hep-th* s pomočjo knjižnice *Prefuse*.

(4) kako velika omrežja lahko vizualiziramo.

Pri prvi točki, kjer smo pogledali kako ročno ustvarimo graf, ni bilo večjih težav. Knjižnice programskega jezika Java - *Gephi*, *JUNG* in *Prefuse*, so potrebovale več nastavitev, vendar so nemoteno izvršile vse navedene ukaze. Pri knjižnici *Prefuse* je bilo zelo specifično dejstvo, da je bilo najprej potrebno ustvariti tabelo, ki je vsebovala vse podatke o vozliščih in povezavah. Pri ostalih knjižnicah programskega jezika Python - *NetworkX* in *igraph*, so bile razlike skorajda neopazne. Najprej je bilo potrebno ustvariti graf in nato direktno dodajati vozlišča in povezave.

Druga točka je vsebovala možnosti knjižnic pri spreminjanju lastnosti komponent grafa. Pri knjižnicah *NetworkX* in *igraph* je to omogočeno z enim samim ukazom, kjer smo vse spremembe podali funkciji za risanje. Pri knjižnicah programskega jezika Java pa ni bilo tako enostavno. Potrebno je bilo vključiti različne razrede, dodati funkcije ali uporabiti zunanje razrede. Pri knjižnici *Prefuse* je bilo pri vsaki vizualizaciji celo obvezno definirati barve, velikosti, debeline itd. Prednost programskega jezika Python in njegovih knjižnic je bila v tem primeru očitna.

Pri tretjemu primerjalnemu kriteriju smo zasledili več razlik med knjižnicami. Knjižnici *NetworkX* in *igraph* sta lahko uporabili pri dodajanju atributov podatkovne strukture programskega jezika Python, seznam ali slovar. Obe knjižnici sta bili tudi v tem koraku dosti podobni. Knjižnica *Prefuse* ima omogočeno tabelo z atributi, ki jih naknadno dodajamo grafu. Knjižnica *JUNG* je na tem področju zelo omejena. Vse knjižnice so imele tudi nekaj skupnega:

dodajanje imen oznak vozlišč in/ali povezav je bilo potrebno že ob samem ustvarjanju le-teh.

Rezultati četrte točke so najbolj zanimivi in raznoliki. Prva in najbolj očitna razlika je pri vizualizaciji majhnega omrežja. Knjižnici *NetworkX* in *Prefuse* sta izrisala omrežje na enak način, a z napako, namreč ukaz knjižnice *NetworkX* za branje podatkov v formatu Pajek je naše podatke prebral kot seznam povezav. Knjižnica *igraph* pa je potrebovala dodaten poseg v podatke za pravilen prikaz usmerjenega omrežja. Pri vizualizaciji srednjega in velikega omrežja je bil prikaz nejasen pri knjižnicah, kjer ni bilo obvezno navesti vrste postavitev. Knjižnice *NetworkX*, *igraph* in *Prefuse* so uporabile algoritem postavitve, medtem ko je *JUNG* poskusil sestaviti okroglo obliko omrežja. Pri knjižnici *Gephi* sta omrežji zgledala najbolj neoblikovano. Največji problem pri tej točki je bilo branje podatkov. Knjižnici *NetworkX* in *Prefuse* nista uspeli narisati največjega omrežja zaradi pomanjkanja spomina. Paket *Prefuse* nima omogočenega branja datotek Pajek, zato je bilo potrebno omrežja pretvoriti v format GML.

V tabeli 5.1 so predstavljene glavne točke, pri katerih se knjižnice za vizualizacijo omrežij razlikujejo. Prva stvar je enostavnost razumevanja knjižnice in sintakse. Pri tem nedvomno zmagata *NetworkX* in *igraph*, prednost pred ostalimi knjižnicami v programskem jeziku Java ima tudi *JUNG*. Drugi stolpec primerjalne tabele predstavlja dokumentacijo, ki je na voljo za vsako izmed knjižnic. Nobena knjižnica ni imela slabe dokumentacije, vendar smo ocenili dokumentacijo paketa *NetworkX* za odlično. Knjižnica je tudi najbolj primerna za začetnike. Tretji, četrti in peti stolpec se nanašajo na vizualizacijo, in sicer avtomatsko prilagajanje velikost vozlišč, branje formata Pajek in uspešnost risanja omrežja z več kot 80000 vozlišč.

Tabela 5.1: Primerjava knjižnic.

Knjižnica	Dokumentacija	Enostavnost	Avt. prilagajanje	format Pajek	80000 vozlišč
Gephi	+	-	-	+	+
NetworkX	++	+	-	+	-
igraph	+	+	+	+	+
JUNG	+	+	-	+	+
Prefuse	+	-	+	-	-

Poglavje 6

Zaključek

Vizualizacija podatkov ima velik pomen za razumevanje omrežij. Za analizo in prikaz omrežij je na voljo veliko programov in knjižnic. Odvisno od tega, kaj potrebujemo, lahko izbiramo med različnimi vrstami programov ali programskih jezikov. Nekatera orodja imajo vgrajene funkcije za specifične domene podatkov - družbeno, biološko, računalniško, medicinsko itd. Orodje izberemo glede na podatke, ceno, možnosti dopolnjevanja in integriranja v lasten projekt, naše znanje programiranja, razpoložljivost.

V okviru diplomskega dela smo pregledali in primerjali pet knjižnic za vizualizacijo omrežij: *Gephi*, *NetworkX*, *igraph*, *JUNG* in *Prefuse*. Pri primerjavi smo pozornost namenili ročnemu ustvarjanju in spreminjanju komponent grafa kot tudi vizualizaciji omrežij različnih velikosti. Pregledali smo bistvene značilnosti vsake od knjižnic in dokumentacijo, ki je na voljo.

Ugotovili smo, da je glavna prednost knjižnic programskega jezika Python njihova enostavnost, pri čemer je *NetworkX* najbolj primerna za uporabnike, ki se prvič soočajo z analizo in vizualizacijo omrežij. Kar se tiče ustvarjanja in vizualizacijo grafov, se je knjižnica *igraph* izkazala za najbolj prilagodljivo. Vsak izmed petih paketov je potreboval vključitev dodatnih knjižnic za dejanski prikaz omrežja. Pri vizualizaciji sta se najbolj izkazali knjižnici *Matplotlib* in *pycairo* programskega jezika Python. Največ problemov so knjižnice imele pri branju velikih količin podatkov. Najbolje je branje uspelo paketom *Gephi*, *igraph* in *JUNG*, medtem ko drugi dve knjižnici nista uspeli pravilno prebrati podatke o majhnem omrežju. Pri tem pa je bil *Prefuse* edini, ki ni znal prebrati formata Pajek. Pri največjih omrežjih brez dodatne izbire algoritma postavitve ali vključitve možnosti povečevanja slike praktično ni bilo mogoče iz vizualizacij ničesar razbrati. Nobena izmed knjižnic ni optimizirana za 64 bitni operacijski sistem Windows, kar lahko močno vpliva na fleksibilnost in

sposobnost učinkovite vizualizacije podatkov.

V bodoče bi bilo zanimivo primerjati večje število knjižnic in programov za vizualizacijo omrežij. Prav tako bi lahko različna orodja primerjali še glede na časovne zahtevnosti algoritmov za vizualizacijo ter količino pomnilnika, ki ga potrebujejo za vizualizacijo omrežij različnih velikosti. Knjižnice pa bi lahko primerjali tudi glede na učinkovitost pri filtriranju omrežja, ki nam omogoča podrobnejšo analizo gruč oziroma izbranih delov omrežja.

Literatura

[1] M. E. J. Newman, "The Structure and Function of Complex Networks", v zborniku *SIAM Review*, Vol. 45, No. 2, str. 167-256, 2003

[2] M. E. J. Newman, "Networks: An Introduction", Oxford University Press, 2010

[3] Quang Vinh Nguyen, Mao Lin Huang, "Large Scale Network Analysis with Interactive Visualisation", Sixth International Conference on Computer Graphics, Imaging and Visualization, 2009

[4] Graph Layout as Optimization. Dostopno na:
`www.csse.monash.edu.au/ berndm/CSE460/Lectures/cse460-7.pdf`

[5] Risanje grafov. Dostopno na:
`http://en.wikipedia.org/wiki/Graph_drawing`

[6] Social network analysis software. Dostopno na:
`http://en.wikipedia.org/wiki/Social_network_analysis_software`

[7] Cyram, "Unleashing Hidden Power of Network", NetMiner, Social Network Analysis Software. Dostopno na:
`http://www.netminer.com/index.php`

[8] NetMiner, Social Network Analysis Software. Dostopno na:
`http://www.netminer.com/index.php`

[9] Borgatti, S.P., M.G. Everett, L.C. Freeman. UCINET 6.0 Version 1.00. Natick: Analytic Technologies. 1999. Dostopno na:
`https://sites.google.com/site/ucinetsoftware/home`

[10] Borgatti, S.P., NetDraw Software for Network Visualization. Analytic Technologies: Lexington, KY. 2002. Dostopno na:
`https://sites.google.com/site/netdrawsoftware/home`

[11] Download Pajek. Dostopno na:
http://pajek.imfm.si/doku.php?id=download

[12] V. Batagelj, A. Mrvar, "Pajek Pajek and Pajek-XXL, Programs for Analysis and Visualization of Very Large Networks, Reference Manual", Ljubljana, 2013. Dostopno na:
http://pajek.imfm.si/doku.php

[13] S. Korenjak-Černe, N. Kejžar, V. Batagelj, "Network Analysis of Works on Clustering and Classification from Web of Science", Ljubljana.

[14] Research at HP Labs, Information Dynamics Lab, Research Areas, Zoomgraph. Dostopno na:
http://www.hpl.hp.com/research/idl/projects/graphs/zoom.html

[15] GUESS: The Graph Exploration System. Dostopno na:
http://graphexploration.cond.org/

[16] ORA. Dostopno na:
http://www.casos.cs.cmu.edu/projects/ora/

[17] Kathleen M. Carley, Jeff Reminga, "ORA: Organization Risk Analyzer, CASOS Technical Report", 2004

[18] Cytoscape. Dostopno na:
http://www.cytoscape.org/

[19] Mark S. Handcock, David R. Hunter, Carter T. Butts, Steven M. Goodreau, Martina Morris, statnet: Software tools for the Statistical Modeling of Network Data, 2003. Dostopno na:
http://statnetproject.org

[20] SNAP, Stanford Network Analysis Platform. Dostopno na:
http://snap.stanford.edu/index.html

[21] Graphviz, Graph Visualization Software, Dostopno na:
http://graphviz.org/About.php

[22] Tulip, Data Visualization Software. Dostopno na:
http://tulip.labri.fr/TulipDrupal/

[23] D. Auber, "Tulip an information visualization software for huge graphs", LaBRI-UniversitÃl' Bordeaux, France, 2002

[24] Tulip Python 4.4.0 documentation. Dostopno na:
tulip.labri.fr/Documentation/4_4/tulip-python/html/index.html

[25] SocNetV, Social Networks Visualizer. Dostopno na:
http://socnetv.sourceforge.net/

[26] Ulrik Brandes, Dorothea Wagner, "Visone, Analysis and Visualization of Social Networks", University of Passau, University of Konstanz, Germany.

[27] Bastian M., Heymann S., Jacomy M. Gephi: an open source software for exploring and manipulating networks. International AAAI Conference on Weblogs and Social Media. 2009. Dostopno na:
https://gephi.org/

[28] Aric Hagberg, Dan Schult, Pieter Swart. NetworkX, 2004-2012. Dostopno na:
http://networkx.github.io/

[29] igraph library. Gabor Csardi, Tamas Nepusz, 2003-2012. Dostopno na:
http://igraph.sourceforge.net/

[30] JUNG, Java Universal Network/Graph Framework. 2003-2010. Dostopno na:
http://jung.sourceforge.net/

[31] The prefuse visualization toolkit. Dostopno na:
http://prefuse.org/

[32] Vladimir Batagelj, Andrej Mrvar: Pajek datasets. 2006. Dostopno na:
http://vlado.fmf.uni-lj.si/pub/networks/data/

[33] Valentina Hlebec: Recall Versus Recognition: Comparison of the Two Alternative Procedures for Collecting Social Network Data. Developments in Statistics and Methodology. (A. Ferligoj, A. Kramberger, editors) Metodološki zvezki 9, FDV, Ljubljana, 1993, p. 121-129.

[34] Beebe, N.H.F.: Nelson H.F. Beebe's Bibliographies Page. 2002. Dostopno na:
http://www.math.utah.edu/ beebe/bibliographies.html

[35] Jones, B., Computational Geometry Database, Februar, 2002. Dostopno na:
http://compgeom.cs.uiuc.edu/~jeffe/compgeom/biblios.html

[36] KDD Cup 2003.
http://www.cs.cornell.edu/projects/kddcup/index.html

[37] Open access to 920,588 e-prints in Physics, Mathematics, Computer Science, Quantitative Biology, Quantitative Finance and Statistics. Dostopno na:
http://arxiv.org/

[38] WordNet, A lexical database of English. Dostopno na:
http://wordnet.princeton.edu/

[39] NetBeans Platform. Dostopno na:
https://netbeans.org/features/platform/index.html